Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

FACTORES A TOMAR EN CUENTA EN EL DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE *SOFTWARE* EN LA MUNICIPALIDAD DE GUATEMALA

Edgar Romeo Salazar Vásquez
Asesorado por el Ing. Ludwin Antonio Rodríguez Tánchez

Guatemala, octubre de 2011

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

FACTORES A TOMAR EN CUENTA EN EL DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE *SOFTWARE* EN LA MUNICIPALIDAD DE GUATEMALA

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

EDGAR ROMEO SALAZAR VÁSQUEZ
ASESORADO POR EL ING. LUDWIN ANTONIO RODRÍGUEZ TÁNCHEZ

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, OCTUBRE DE 2011

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Ing. Miguel Ángel Dávila Calderón
VOCAL IV	Br. Juan Carlos Molina Jiménez
VOCAL V	Br. Mario Maldonado Muralles
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Juan Álvaro Díaz Ardavin
EXAMINADOR	Ing. Oscar Alejandro Paz Campos
EXAMINADOR	Ing. César Augusto Fernández Cáceres
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

FACTORES A TOMAR EN CUENTA EN EL DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE *SOFTWARE* EN LA MUNICIPALIDAD DE GUATEMALA

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha agosto de 2010.

Edgar Romeo Salazar Vásquez

Guatemala 05 de mayo de 2011

Ingeniero
Carlos Alfredo Azurdia Morales
Facultad de Ingeniería
Universidad de San Carlos de Guatemala
Presente

Ingeniero Azurdia:

Por medio de la presente informo a usted, que he procedido a revisar el trabajo de graduación elaborado por el estudiante **EDGAR ROMEO SALAZAR VÁSQUEZ,** con carné 1999-11894 de la carrera de Ingeniería en Sistemas cuyo título es: **"FACTORES A TOMAR EN CUENTA EN EL DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE SOFTWARE EN LA MUNICIPALIDAD DE GUATEMALA".**

Considero que el trabajo presentado por el estudiante Salazar Vásquez, ha sido desarrollado cumpliendo con los requisitos reglamentarios, por lo que doy mi aprobación y solicito el trámite correspondiente.

Sin otro particular me es grato suscribirme de usted, muy respetuosamente.

Ing. Ludwin Antonio Rodríguez Tánchez
Asesor
Colegiado 6470

Ing. Ludwin A. Rodriguez ...
Ingeniero en Sistemas de Información
y Ciencias de la Computación
Colegiado Activo 6470

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 1 de Junio de 2011

Ingeniero
Marlon Antonio Pérez Turk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **EDGAR ROMEO SALAZAR VÁSQUEZ,** carné **1999-11894,** titulado: **"FACTORES A TOMAR EN CUENTA EN EL DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE SOFTWARE EN LA MUNICIPALIDAD DE GUATEMALA",** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
INGENIERIA EN CIENCIAS Y SISTEMAS
COORDINACION TESIS

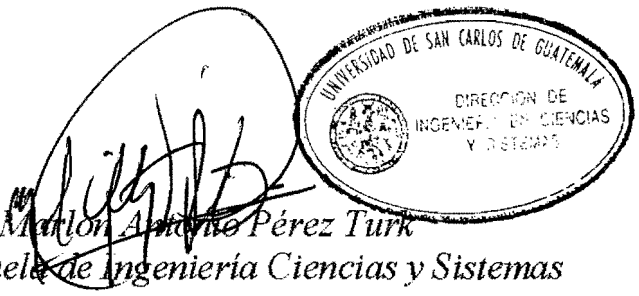
ESCUELA DE CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
TEL: 24767644

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, de trabajo de graduación titulado ***"FACTORES A TOMAR EN CUENTA EN EL DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE SOFTWARE EN LA MUNICIPALIDAD DE GUATEMALA",*** *presentado por el estudiante EDGAR ROMEO SALAZAR VÁSQUEZ, aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"



Ing. Marlon Antonio Pérez Turk
Director, Escuela de Ingeniería Ciencias y Sistemas

Guatemala, 21 de octubre 2011

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

DTG. 422.2011

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **FACTORES A TOMAR EN CUENTA EN EL DESARROLLO E IMPLEMENTACIÓN DE UN SISTEMA DE *SOFTWARE* EN LA MUNICIPALIDAD DE GUATEMALA**, presentado por el estudiante universitario **Edgar Romeo Salazar Vásquez**, autoriza la impresión del mismo.

IMPRÍMASE:

Ing. Murphy Olympo Paiz Recinos
Decano

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, 21 de octubre de 2011.

/gdech

ACTO QUE DEDICO A:

Dios	Por ser mí guía, mi luz y fortaleza; porque ha estado conmigo siempre y me ha guardado de todo mal.
Mis padres	Israel Salazar y Zoila Vásquez, gracias por sus sabios consejos, cariño, motivación, apoyo incondicional y por hacer de mí un hombre de bien.
Mis hermanos	Norma, Marlon y Leidy, Con quienes comparto este logro, gracias por su apoyo.
Mis amigos	Obed Mazariegos, Julio González, César Rivas, por su amistad y apoyo que en su momento me brindaron.
Facultad de Ingeniería	Por darme la formación académica, gracias ingeniería

AGRADECIMIENTOS A:

Ing. Ludwin Antonio Rodríguez Tánchez	Por su asesoría en la realización de este trabajo
Inga. Sonia Castañeda	Por toda la orientación y apoyo que amablemente me concedió, durante la elaboración de este trabajo
Personal de las áreas de informática en la Municipalidad de Guatemala	Por brindarme todo el apoyo necesario y hacer realidad el presente trabajo de graduación
Licda. Sandra Batres	Por la revisión de este trabajo y apoyo que amablemente me brindo.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

Atómicas	No puede ser descompuesto.
Back-Out	Refuerzo.
BD	Base de datos.
CAB	*Change advisory board.*
Causahabiente	Persona física o jurídica que ha sustituido a otra.
Change advisory board	Comité de cambios. Personal que asesora al gerente de cambios en la valoración, priorización y planificación de los cambios. Este comité está formado por representantes de todas las áreas del proveedor de servicios de tecnología informática, del negocio, y proveedores externos.
CI	*Configure ítem.*
Cliente-servidor	Arquitectura que consiste en un cliente que realiza peticiones a otro programa (servidor) que le da respuesta.
CMDB	*Configuration management database.*

Colectivo	Grupo de personas que comparten y están motivados por un mismo tema u objetivo de interés.
Compeler	Obligar a alguien, con fuerza o con el poder de la autoridad a realizar una cosa.
Configuration Item	Elemento de configuración. Cualquier componente que necesite ser gestionado con el objeto de proveer un servicio de tecnología de información.
Configuration management database	Base de datos de gestión de la configuración. Base de datos usada para almacenar registros de configuración durante todo su ciclo de vida.
Entidad	Objeto del que se recoge información de interés para el sistema.
Framework	Es una estructura de soporte definida en la cual otro proyecto de *software* puede ser organizado y desarrollado.
Hito	Punto de control de objetivo intermedio antes de que el proyecto finalice.
IBM	*International bussiness machines.*

Infraestructura	Conjunto de elementos o servicios que se consideran necesarios para la creación y funcionamiento de una organización, sistema, etc.
ISR	Impuesto sobre la renta.
ITIL	*Information technology infrastructure library.*
IVA	Impuesto al valor agregado.
Modelo Entidad-Relación	Herramienta para el modelado de datos de un sistema o programa.
Módulo	Parte de un programa o sistema.
N-capas	Separación de la lógica del negocio de la lógica de diseño.
Outrsorcing	Delegar funciones o actividades internas de una empresa a un proveedor externo.
PC	*Personal computer.*
PDA	*Personal digital assistant.*
Prefacio	Prólogo o introducción.

Prueba de Caja Blanca	Centra en los detalles procedimentales del *software*, su diseño está fuertemente ligado al código.
QA	*Quality assurance.*
RAD	*Rapid application development.*
Recíproco	Igual en la correspondencia de uno a otro.
Refactorizar	Es una técnica de la ingeniería de *software* para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.
Relación	Asociación de dos o más entidades.
Request for change	Petición de cambio. Propuesta formal para que se realice un cambio. Incluye detalles del cambio propuesto, y puede registrarse en papel o electrónicamente.
RFC	*Request for change.*
Rugby	Deporte de contacto en equipo nacido en Inglaterra.
RUP	*Rational modeling language.*

Stakeholders	Quienes pueden afectar o son afectados por las actividades de una empresa.
TI	Tecnología informática.
UML	*Unified modeling language.*
Unilateral	Se refiere a una sóla parte o lado.
Workflow	Es el conjunto de actividades o tareas realizadas en secuencia o en paralelo.
XP	*eXtreme programming.*

RESUMEN

En la Municipalidad de Guatemala existen departamentos en los que se desarrolla el *software,* entre ellos están: informática de la Municipalidad de Guatemala, Entidad Metropolitana Reguladora de Transporte y Transito (EMETRA), Empresa Municipal de Agua (EMPAGUA), Catastro y licencias de la construcción. Los departamentos cuentan con su propio personal y cada uno utiliza diferentes formas para desarrollar *software*.

Para ayudar a mejorar el rendimiento y satisfacción de los clientes en la Municipalidad de Guatemala se revisó como desarrollan el *software* y con base a la información se realiza un análisis para verificar qué factores son necesarios, para mejorar el desarrollo del *software*, que cumpla con las necesidades del cliente.

Los factores necesarios para desarrollar *software* en los departamentos de la Municipalidad de Guatemala se clasificaron en 4 fases: funcionamiento y mantenimiento, inicio, desarrollo y aplicación.

En la fase de funcionamiento y mantenimiento, se encuentran los siguientes factores: reglas del negocio, metodología para el modelado de procesos y actividades y contratos; en la fase de inicio: toma de requerimientos, usuarios y roles, requerimientos de *hardware*, requerimientos de *software*, definición de estándares; en la fase de desarrollo: tipos de pruebas, pruebas de estrés, pruebas unitarias, herramientas para tipos de pruebas, trabajo en equipo; en la fase de aplicación: manual de usuario, manual técnico, gestión de cambios y satisfacción del cliente.

La metodología utilizada para el desarrollo de los factores es *Rational Unified Process (*RUP) y se realiza la relación que existe entre los entregables del RUP y los factores necesarios para mejorar el desarrollo del *software* en los departamentos de la Municipalidad de Guatemala.

OBJETIVOS

General

Generar una guía de consulta para que las empresas que desarrollan *software* puedan utilizarla al momento de realizar un sistema y determinen los factores necesarios para que el desarrollo de sus sistemas sea exitoso.

Específicos

1. Determinar cuáles son los factores claves para el desarrollo de un sistema informático.

2. Ayudar a las empresas a entender, cuáles son los factores importantes en el desarrollo de un sistema de *software*.

INTRODUCCIÓN

Las empresas que se dedican al desarrollo de *software* enfrentan cada día la posibilidad de mal funcionamiento los proyectos que desarrollan, esto hace que muchas de ellas lleguen al fracaso; para solucionar este problema, se pretende indicar, cuáles son los factores que se tienen que considerar al momento que se desarrolla un nuevo sistema, y con esto, se puede disminuir el fracaso de los proyectos y aumentar la confiabilidad de las personas que necesitan un sistema.

El desarrollo de un nuevo sistema conlleva una gran variedad de aspectos a determinar, pero de ellos, unos son principales y otros secundarios; lo que se pretende es mostrar a las empresas, cuáles son los que realmente se requieren y se necesitan, para que un proyecto finalice a tiempo y obtener la satisfacción de los clientes.

1. METODOLOGÍAS ÁGILES

“A principios de la década del ’90, surgió un enfoque que fue bastante revolucionario para su momento, ya que estaba en contra de toda creencia de que mediante procesos altamente definidos, se iba a lograr obtener *software* en tiempo, costo y con la requerida calidad.

El enfoque fue planteado por primera vez por Martin y se dio a conocer en la comunidad de Ingeniería de *Software* con el nombre de RAD o *Rapid Application Development*. RAD consistía en un entorno de desarrollo altamente productivo, en el que participaban grupos pequeños de programadores utilizando herramientas que generaban código en forma automática tomando como entradas sintaxis de alto nivel. En general, se considera que este fue uno de los primeros hitos en pos de la agilidad en los procesos de desarrollo.

La historia de las metodologías ágiles y su apreciación como tales en la comunidad de la ingeniería de *software*, tiene sus inicios en la creación de una de las metodologías utilizada como arquetipo: *eXtreme Programming* XP, que nace de la mente de *Kent Beck*, tomando ideas recopiladas junto a *Ward Cunningham*.

Durante 1996, *Beck* es llamado por *Chrysler* como asesor del proyecto *Chrysler Comprehensive Compensation payroll system*. Dada la poca calidad del sistema que se estaba desarrollando, *Beck* decide tirar todo el código y empezar de cero utilizando las prácticas que él había definido a lo largo del tiempo. El sistema que administra la liquidación de aproximadamente 10.000 empleados y consiste de 2.000 clases y 30.000 métodos, es puesto en operación en mayo de 1997, casi respetando el calendario propuesto. Como consecuencia del éxito de dicho proyecto *Kent Beck* dio origen a XP iniciando el movimiento de metodologías ágiles al que se anexarían otras metodologías surgidas mucho antes que el propio *Beck* fuera convocado por *Chrysler*.

Es así como que este tipo de metodologías fueron inicialmente llamadas “metodologías livianas”, sin embargo, aún no contaban con una aprobación pues se le consideraba por muchos programadores como meramente intuitiva. Luego, con el pasar de los años, en febrero de 2001, tras una reunión celebrada en Utah, Estados Unidos, nace formalmente el término “ágil” aplicado al desarrollo de *software*.

En esta misma reunión participa un grupo de 17 expertos de la industria del *software*, incluyendo algunos de los creadores o impulsores de metodologías de *software,* con el objetivo de esbozar los valores y principios que deberían permitir a los equipos desarrollar *software* rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de *software* tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.

Tras esta reunión se creó *The Agile Alliance*, una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de *software* y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el manifiesto ágil, un documento que resume la filosofía ágil".[1]

1.1. Metodologías ágiles versus metodologías tradicionales

En la tabla I se muestran las principales diferencias de las metodologías ágiles con respecto de las metodologías tradicionales.

Tabla I. **Metodologías ágiles versus metodologías tradicionales**

Metodologías ágiles	Metodologías tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparadas para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo).	Impuestas externamente.
Proceso menos controlados, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más artefactos.
Pocos roles.	Más roles.
Menos énfasis en la arquitectura del *software*.	La arquitectura del *software* es esencial y se expresa mediante modelos.

Fuente: CALDERÓN, Amaro. Metodologías ágiles. p. 9.

[1] CALDERÓN, Amaro. "Metodologías Ágiles". Universidad Nacional de Trujillo. 2007. http://www.seccperu.org/files/Metodologias%20Agiles.pdf (7 de agosto de 2010).

"Tener metodologías diferentes para aplicar de acuerdo con el proyecto que se desarrolle resulta una idea interesante. Estas metodologías pueden involucrar prácticas tanto de metodologías ágiles como de metodologías tradicionales. De esta manera se podría tener una metodología para cada proyecto, la problemática sería definir cada una de las prácticas y en el momento preciso explicar parámetros para saber cuál usar.

Es importante tener en cuenta que el uso de un método ágil no es para todos. Sin embargo, una de las principales ventajas de los métodos ágiles es su peso inicialmente ligero y por eso las personas que no estén acostumbradas a seguir procesos encuentran estas metodologías bastante agradables".[2]

1.2. *Rational Unified Process* (RUP)

"Es un proceso de desarrollo de *software* y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización".[3]

[2] CALDERÓN, Amaro. "Metodologías Ágiles", Universidad Nacional de Trujillo, 2007, http://www.seccperu.org/files/Metodologias%20Agiles.pdf (7 de agosto de 2010).
[3] Wikipedia, "Proceso Unificado de *Rational*", http://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational (20 de agosto 2010).

1.2.1. Características y beneficios de RUP

"No existen dos proyectos de desarrollo de *software* que sean iguales. Cada uno tiene prioridades, requerimientos, y tecnologías muy diferentes. Sin embargo, en todos los proyectos, se debe minimizar el riesgo, garantizar la predictibilidad de los resultados y entregar *software* de calidad superior a tiempo. *Rational Unified Process* o RUP que es una plataforma flexible de procesos de desarrollo de *software* que ayuda suministrando guías consistentes y personalizadas de procesos para todo el equipo de proyecto.

- Las mejores prácticas más probadas de la industria: son las mejores prácticas de desarrollo adoptadas en proyectos a nivel mundial y enseñadas como parte del pensum en cientos de universidades, la metodología RUP se convirtió rápidamente en el estándar para el proceso de desarrollo en la industria de *software*.

- Proceso hecho práctico: diferente que otras metodologías comerciales, la plataforma RUP hace que el proceso sea práctico con bases de conocimiento y guías para ayudar en el despegue de la planificación del proyecto, integrar rápidamente a los miembros del equipo y poner en acción el proceso personalizado.

- Se adapta a las necesidades de los proyectos: sólo la plataforma RUP proporciona un *framework* de proceso configurable que permite seleccionar e implantar los componentes específicos de proceso necesarios para proporcionar un proceso consistente y customizado para cada equipo y proyecto.

Una de las mejores prácticas centrales de RUP es la noción de desarrollar iterativamente. *Rational Unified Process* organiza los proyectos en términos de disciplinas y fases, consistiendo cada una en una o más iteraciones. Con esta aproximación iterativa, el énfasis de cada *workflow* variará a través del ciclo de vida. La aproximación iterativa ayuda a mitigar los riesgos en forma temprana y continua, con un progreso demostrable y frecuentes *releases* ejecutables".[4]

1.2.2. Ciclo de vida del RUP

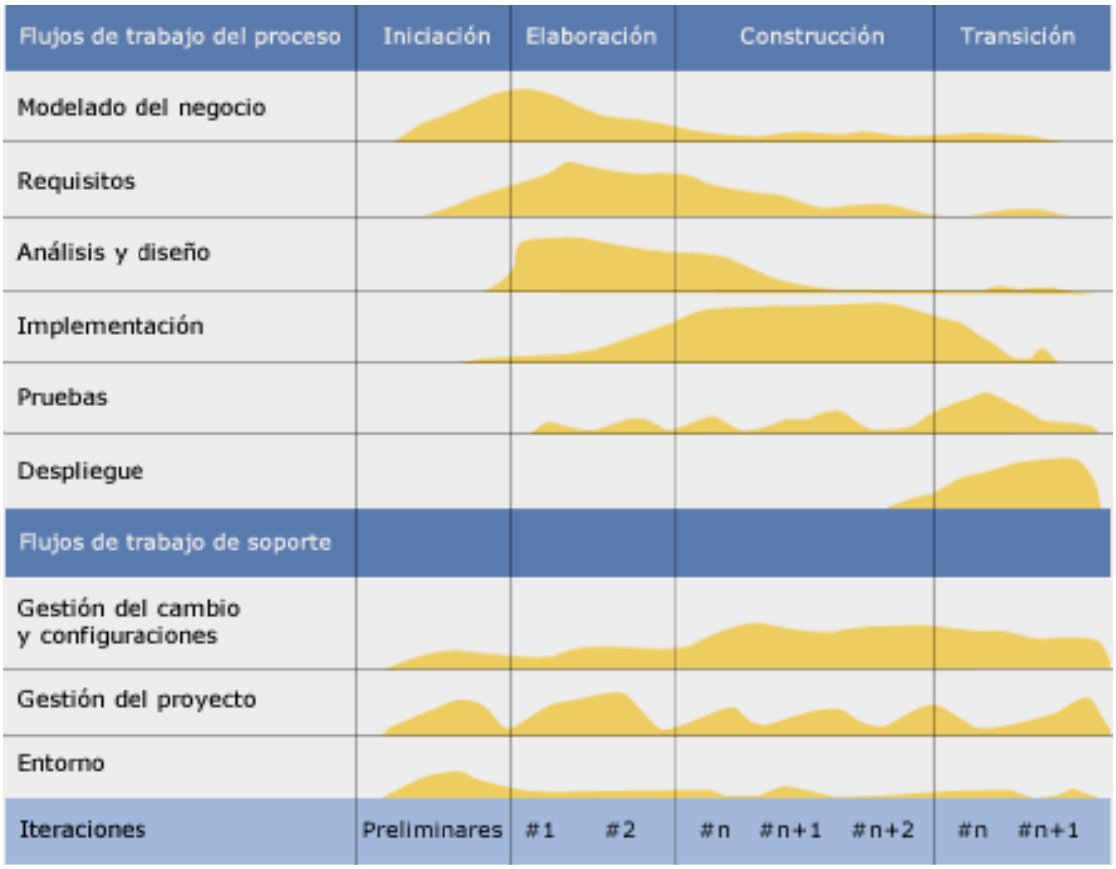
"El ciclo de vida de RUP, está dividido en 4 fases: inicio, elaboración, construcción y transición, que corresponden a los 4 hitos principales de RUP. El RUP está dividido en principios clave. Cada uno de ellos corresponde a distintos aspectos del desarrollo de *software* que generalmente requieren habilidades específicas; esto se refleja en los roles y las actividades definidas para cada principio.

Cada fase cambia el foco del equipo de trabajo para alcanzar cada uno de los hitos y es llevada a cabo en forma iterativa. Esto quiere decir que, la fase se fragmenta en pequeños proyectos que recorren todas las disciplinas y producen un ejecutable".[5]

En la figura 1, se describen los flujos del trabajo y las fases necesarias para el ciclo de vida del RUP.

[4] GSI. "*Rational Unified Process*", 2007, http://www.rational.com.ar/herramientas/rup.html (21 de agosto 2010).
[5] Itera. "Marco de Referencia", *Rational Unified Process*, 2010, http://www.iteraprocess.com/index.php?option=com_content&task=view&id=18&Itemid=42&limit=1&limitstart=1 (21 de agosto 2010).

Figura 1. **Ciclo de vida del RUP**



Fuente: wikipedia. Proceso unificado de *rational.* http://es.wikipedia.org/wiki/Proceso _Unificado_de_Rational. Fecha de consulta: 20 de agosto 2010.

A continuación se describe cada una de las fases del RUP.

- "Inicio: alcanzar un acuerdo entre todos los interesados respecto a los objetivos del ciclo de vida para el proyecto, generando el ámbito del proyecto, el caso de negocio, síntesis de arquitectura posible y el alcance del proyecto".[6] Los objetivos de esta fase son:

[6] Itera. "Marco de Referencia", *Rational Unified Process*, 2010, http://www.iteraprocess.com/index.php?option=com_content&task=view&id=18&Itemid=42&limit=1&limitstart=1 (21 de agosto 2010).

- “Establecer el ámbito del proyecto y sus límites
- Encontrar los casos de uso críticos del sistema, los escenarios básicos que definen la funcionalidad
- Mostrar al menos una arquitectura, candidata para los escenarios principales
- Estimar el costo en recursos y tiempo de todo el proyecto
- Estimar los riesgos, las fuentes de incertidumbre

Los productos de la fase de inicio deben ser:

- Visión del negocio
- Modelo de casos de uso
- Especificaciones adicionales
- Glosario
- Lista de riesgos y planes de contingencia
- El caso de uso del negocio
- Prototipos exploratorios para probar conceptos o la arquitectura candidata
- Plan de iteración para la primera iteración de la fase de elaboración
- Plan de fases

No todos los productos son obligatorios, ni deben completarse al 100%, debemos tener en cuenta los objetivos de la fase de inicio”.[7]

[7] MARTÍNEZ, Alejandro; MARTÍNEZ, Raúl. “Guía a *Rational Unified Process*” http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia%20RUP.pdf (22 de octubre 2010).

Para la realización de los proyectos en los departamentos que desarrollan *software* en la Municipalidad de Guatemala, en esta fase se encuentran los siguientes factores, los que son considerados claves en el inicio del desarrollo de los sistemas municipales:

- Reglas del negocio
- Metodología para el modelado de procesos y actividades
- Contratos

Los factores anteriores se describen a detalle en el capítulo 2, existen otros factores que también se toman en cuenta para otras empresas, pero en la Municipalidad de Guatemala se que los importantes son los mencionados anteriormente.

- "Elaboración**:** establecimiento de la línea base para la arquitectura del sistema y proporcionar una base estable para el diseño y el esfuerzo de implementación de la siguiente fase, mitigando la mayoría de los riesgos tecnológicos".[8] Los objetivos de la fase de elaboración son:

  - "Definir, validar y cimentar la arquitectura
  - Completar la visión
  - Crear un plan fiable para la fase de construcción
  - Demostrar que la arquitectura propuesta soportar la visión con un costo razonable y en un tiempo razonable

[8] Itera. "Marco de Referencia", *Rational Unified Process*, 2010, http://www.iteraprocess.com/index.php?option=com_content&task=view&id=18&Itemid=42&limit=1&limitstart=1 (21 de agosto 2010).

Al finalizar la fase se deben de obtener los siguientes productos:

- Un modelo de caso de uso completo al menos un 80%
- Requisitos adicionales
- Descripción de la arquitectura *software*
- Un prototipo ejecutable de la arquitectura
- Lista de riesgos y casos de negocio revisados
- Plan de desarrollo del proyecto
- Un caso de desarrollo actualizado que especifica el proceso a seguir
- Posiblemente un manual de usuario preliminar

En la forma de aproximarse a esta fase se trata de abarcar todo el proyecto con la profundidad mínima. Sólo se debe profundizar en los puntos críticos de la arquitectura o riesgos importantes".[9]

Los departamentos de informática de la Municipalidad de Guatemala, se encuentran con el problema de los usuarios, para minimizar los riesgos de fracasos en la toma de requerimientos, poco interés de los usuarios, problemas de *software* y *hardware*, etc., se propone tomar en cuenta los siguientes factores:

- Toma de requerimientos
- Usuarios y roles
- Requerimientos de *hardware*
- Requerimientos de *software*
- Definición de estándares

[9] MARTÍNEZ, Alejandro; MARTÍNEZ, Raúl. "Guía a *Rational Unified Process*" http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia%20RUP.pdf (22 de octubre 2010).

Los factores anteriores se describen a detalle en el capítulo 3, existen otros factores que también se toman en cuenta para otras empresas, pero en la Municipalidad de Guatemala se que los importantes son los mencionados anteriormente.

- "Construcción: completar el desarrollo del sistema basado en la línea base de la arquitectura".[10] "La finalidad principal de esta fase es alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones. Durante esta fase todas los componentes, características y requisitos, que no hayan sido hecho hasta ahora, han de ser implementados, integrados y testeados, obteniéndose una versión del producto que se pueda disponer a los usuarios (una versión de prueba). Los objetivos concretos de esta fase son:

  - Minimizar los costos de desarrollo mediante la optimización de recursos y evitando rehacer un trabajo o incluso desecharlo
  - Conseguir una calidad adecuada tan rápido como sea práctico
  - Conseguir versiones funcionales (alfa, beta y otras versiones de prueba) tan rápido como sea práctico

  Los productos de la fase de construcción son:

  - Modelos completos (casos de uso, análisis, diseño, despliegue e implementación)
  - Arquitectura integrada (mantenida y mínimamente actualizada)
  - Riegos presentados mitigados
  - Plan del proyecto para la fase de transición

[10] Itera. "Marco de Referencia", *Rational Unified Process*, 2010, http://www.iteraprocess.com/index.php?option=com_content&task=view&id=18&Itemid=42&limit=1&limitstart=1 (21 de agosto 2010).

- Manual inicial de usuario (con suficiente detalle)
- Prototipo operacional (beta)
- Casos del negocio actualizado"[11]

Para cumplir con las necesidades y requerimientos de los usuarios de la Municipalidad de Guatemala, se deben de tomar en cuenta los siguientes factores en esta fase:

- Tipos de pruebas
- Pruebas de estrés
- Pruebas unitarias
- Herramientas para tipos de pruebas
- Trabajo en equipo

Los factores que son necesarios para realizar la fase de construcción se detallan en el capítulo 4 y se consideran importantes para los departamentos que desarrollan *software* en la Municipalidad de Guatemala.

- "Transición: garantizar que el *software* esté listo para entregarlo a los usuarios".[12] "La finalidad de la fase de transición es disponer el producto a los usuarios finales, para lo que típicamente se requiere desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto y en general tareas relacionadas con el ajuste, configuración, instalación y utilidad del producto. Los objetivos importantes de esta fase son:

[11] MARTÍNEZ, Alejandro; MARTÍNEZ, Raúl. "Guía a *Rational Unified Process*" http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia%20RUP.pdf (22 de octubre 2010).
[12] Itera. "Marco de Referencia", *Rational Unified Process*, 2010, http://www.iteraprocess.com/index.php?option=com_content&task=view&id=18&Itemid=42&limit=1&limitstart=1 (21 de agosto 2010).

- Conseguir que el usuario se valga por sí mismo
- Un producto final que cumpla los requisitos esperados, que funcione y satisfaga suficientemente al usuario

Los productos de la fase de transición son:

- Prototipo operacional
- Documentos legales
- Caso del negocio completo
- Línea base del producto completada y corregida que incluye todos los modelos del sistema
- Descripción de la arquitectura completa y corregida

Las iteraciones de esta fase irán dirigidas a una nueva versión. La complejidad de esta fase depende totalmente de la naturaleza del proyecto, de su alcance y de la organización donde se implementa".[13]

Los departamentos que desarrollan *software* en la Municipalidad de Guatemala le dan poca importancia a la documentación relacionada, a la fase de transición y es por eso que se consideran importantes los siguientes factores, se detallan en el capítulo 5:

- Manual de usuario
- Manual técnico
- Gestión de cambios
- Satisfacción del cliente

[13] MARTÍNEZ, Alejandro; MARTÍNEZ, Raúl. "Guía a *Rational Unified Process*" http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia%20RUP.pdf (22 de octubre 2010).

1.2.3. Relación de los productos del RUP con los factores de desarrollo e implementación de un *software*

En la tabla II se determina la relación que existe entre los productos que se desarrollan en el RUP, los factores que se consideraran importantes en el desarrollo e implementación de los proyectos y los departamentos de tecnología informática TI, que deben utilizar el factor en el desarrollo e implementación de los proyectos en la Municipalidad de Guatemala.

Tabla II. **Relación de productos del RUP, los factores de desarrollo de *software* y los departamentos de TI de la Municipalidad de Guatemala**

Fase	Producto	Factor	Departamento TI
Inicio	Visión del negocio Glosario	Reglas del negocio	Informática Municipal Emetra Empagua Licencias de la construcción Catastro
	Modelado de casos de uso Glosario	Metodología para el modelado de procesos y actividades	Emetra Licencias de la construcción Catastro
	Especificaciones Adicionales Lista de riesgos y planes de contingencia Glosario	Contratos	Emetra
Elaboración	Un modelo de caso de uso completo al menos un 80%	Toma de requerimientos	Emetra Catastro
	Plan de desarrollo del proyecto	Usuarios y roles	Emetra Licencias de la construcción Catastro
	Descripción de la arquitectura *software*	Requerimientos de *hardware*	Informática municipal Emetra Licencias de la construcción Catastro
	Requisitos adicionales	Definición de estándares	Emetra Empagua Licencias de la construcción Catastro
		Requerimientos de *software*	Informática municipal Emetra Empagua Licencias de la construcción Catastro

Continuación tabla II

Construcción	Prototipo operacional (beta) Arquitectura integrada (mantenida y mínimamente actualizada)	Pruebas de estrés	Informática municipal Emetra Empagua Licencias de la construcción Catastro
		Pruebas unitarias	Todos los departamentos realizan pruebas unitarias pero es necesario que realicen la documentación
		Herramientas para tipos de pruebas	Informática municipal Emetra Empagua Licencias de la construcción Catastro
	Modelos completos (casos de uso, análisis, diseño, despliegue e implementación)	Trabajo en equipo	Informática municipal Emetra Empagua Licencias de la construcción Catastro
Transición	Documentos legales	Manual de usuario	Emetra
		Manual técnico	Emetra Catastro
	Prototipo operacional	Satisfacción del cliente	Emetra
	Línea base del producto completada y corregida que incluye todos los modelos del sistema Caso del negocio completo Descripción de la arquitectura completa y corregida	Gestión de cambios	Informática municipal Emetra Empagua Licencias de la construcción Catastro

Fuente: elaboración propia.

1.2.4. Flujo de trabajo del RUP

Tabla III. **Flujo de trabajo del proceso unificado de *rational***

Flujo de Trabajo	Descripción
Modelado del negocio.	Los procesos se modelan utilizando casos de uso del negocio.
Requisitos	Se definen los actores que interactúan con el sistema y se desarrollan casos de uso para modelar los requerimientos del sistema.
Análisis y diseño	Se crea y documenta un modelo del diseño utilizando modelos arquitectónicos, modelos de componentes, modelos de objetos y modelos de secuencia.
Implementación	Se implementan y estructuran en subsistemas los componentes del sistema. La generación automática de código de los modelos del diseño ayuda a acelerar este proceso.
Pruebas	Las pruebas son un proceso iterativo que se llevan a cabo conjuntamente con la implementación. A la finalización de la implementación tienen lugar las pruebas del sistema.
Despliegue	Se crea una *reléase* del proyecto, se distribuye a los usuarios y se instala en su lugar de trabajo.
Gestión de cambios y configuraciones	Este flujo de trabajo de soporte gestiona los cambios del sistema.
Gestión del proyecto	Este flujo de trabajo de soporte gestiona el desarrollo del sistema.
Entorno	Este flujo de trabajo se refiere a hacer herramientas de *software* apropiadas disponibles para los equipos de desarrollo de *software*.

Fuente: SOMMERVILL, Ian. Ingeniería de *software*. p. 78.

1.2.5. Roles del proceso unificado de *rational*

En la tabla siguiente se muestran los roles principales del RUP, descripción de cada rol y en que disciplina intervienen los roles.

Tabla IV. **Roles del proceso unificado de *rational***

Disciplina	Rol	Descripción Rol
Modelado del negocio	Analista del procesos de negocio	Coordina y conduce todos los casos de uso del negocio
	Diseñador del negocio	Detalla uno o varios casos de uso del negocio
Requisitos	Analista de sistemas	Conduce y coordina los requerimientos y los casos de uso, modelando y delimitando la funcionalidad del sistema.
	Especificador de requerimientos	Detalla la especificación de una parte de la funcionalidad del sistema
Análisis y diseño	Arquitecto de *software*	Decide qué tecnologías utilizar
	Diseñador	Define las responsabilidades, las operaciones, los atributos, y las relaciones de una o varias clases y determina cómo serán ajustadas al ambiente de implementación
Implementación	Integrador	Define la integración de las clases
	Implementador	Es responsable de desarrollar y de probar componentes de acuerdo con los estándares adoptados del proyecto para la integración en subsistemas más grandes.
Pruebas	Jefe de pruebas	Tiene la responsabilidad total del éxito de las pruebas
	Diseñador de pruebas	Responsable de definir las técnicas apropiadas, herramientas y pautas para implementar las pruebas requeridas y asegurar su implementación satisfactoria
	Analista de pruebas	Responsable de identificar y definir las pruebas requeridas
	Revisor	Encargado de realizar las pruebas
Despliegue	Jefe de despliegue	Planea la transición del producto a los usuarios y sus documentos
	Documentador técnico, desarrollador de cursos y artista gráfico	Crear materiales detallados para asegurar la implementación exitosa
Gestión del cambio y configuraciones	Jefe de configuración	Configura el entorno de la gerencia de la configuración (CM), políticas y plan.
	Jefe de control de cambios	Supervisa el proceso de control de cambios
Gestión del proyecto	Jefe del proyecto	Asignar recursos, coordinar interacciones con los clientes y los usuarios, mantener al equipo del proyecto centrado en el objetivo correcto.
Entorno	Ingeniero de procesos	Responsable del proceso de desarrollo del *software*
	Especialista en herramientas	Crea las condiciones para el uso de las herramientas

Fuente: CRAIN, Anthony. U*nderstanding* RUP roles. http://www.ibm.com/developerworks/ rational/library/apr05/crain/index.html. Fecha de consulta: 20 de agosto 2010.

1.3. Programación extrema (*extreme programming XP*)

"La programación extrema (XP) es posiblemente el método ágil más conocido y ampliamente utilizado. El nombre fue creado por *Beck*, debido a que el enfoque fue desarrollado utilizando buenas prácticas reconocidas, como el desarrollo iterativo y con la participación del cliente en niveles extremos.

En la programación extrema, todos los requerimientos se expresan como escenarios (llamados historias de usuario), los cuales se implementan directamente como una serie de tareas. Los programadores trabajan en parejas y desarrollan pruebas para cada tarea antes de escribir el código. Todas las pruebas se ejecutan satisfactoriamente cuando el código nuevo se integra al sistema. Existe un pequeño espacio de tiempo entre las entregas del sistema, la figura 2 ilustra el proceso de XP para producir un incremento del sistema que se está desarrollando".[14]

Figura 2. **Ciclo de vida XP**

Fuente: elaboración propia.

[14]SOMMERVILLE, Ian. Ingeniería de *software*(Madrid: *Pearson Education* S.A.,2005), p. 364.

1.3.1. Ciclo de vida XP

El ciclo de vida ideal de XP consiste en seis fases:

- “Exploración: los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo, el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizan en el proyecto. Se prueba la tecnología y se explora las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

- Planificación de la entrega: el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se determinan acuerdos sobre el contenido de la primera entrega, así como un cronograma en conjunto con el cliente. Una entrega debe obtenerse en no más, de tres meses. Esta fase dura unos pocos días.

- Iteraciones: esta fase incluye varias iteraciones sobre el sistema, antes de ser entregado. El plan de entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que es utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, ésto no siempre es posible ya que, es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio).

Al final de la última iteración el sistema está listo para entrar en producción. Los elementos que deben tomarse en cuenta durante la elaboración del plan de la iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

- Producción: la fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

  Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación.

- Mantenimiento: mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento requiere nuevo personal dentro del equipo y cambios en su estructura.

- Muerte del proyecto: es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo"[15].

1.3.2. Prácticas de XP

La programación extrema implica varias prácticas, las prácticas se encuentran resumidas en la Tabla III, que se ajustan a los principios de las metodologías ágiles.

[15] LETELIER, Patricia; PENADÉZ, Mª Carmen. "Metodologías ágiles para el desarrollo de *software*: *eXtreme Programing*(XP)",2004, http://www.willydev.net/descargas/masyxp.pdf(10 de septiembre 2010).

Tabla V. **Prácticas de XP**

Principio o Práctica	Descripción
Planificación incremental	Los requerimientos se registran en tarjetas de historia y las historias a incluir en una entrega se determinan según el tiempo disponible y su prioridad relativa. Los desarrolladores dividen estas historias en tareas de desarrollo. Véase la figura 3 y figura 4.
Entregas pequeñas	El mínimo conjunto útil de funcionalidad que proporcione valor de negocio se desarrolla primero. Las entregas del sistema son frecuentes e incrementalmente añaden funcionalidad a la primera entrega.
Diseño sencillo	Sólo se lleva a cabo el diseño necesario para cumplir los requerimientos actuales.
Desarrollo previamente probado	Se utiliza un sistema de pruebas de unidad automatizada para escribir pruebas para nuevas funcionalidades antes, de que éstas se implementen.
Refactorización	Se espera que todos los desarrolladores refactoricen el código continuamente tan pronto como encuentren posibles mejoras en el código. Ésto conserva el código sencillo y mantenible.
Programación en parejas	Los desarrolladores trabajan en parejas, verificando cada uno el trabajo del otro y proporcionan la ayuda necesaria para hacer siempre un buen trabajo.
Propiedad colectiva	Las parejas de desarrolladores trabajan en todas las áreas del sistema, de modo que no desarrollen islas de conocimientos y todos los desarrolladores posean todo el código. Cualquiera puede cambiar cualquier cosa.
Integración continua	En cuanto acaba el trabajo en una tarea, se integra en el sistema entero. Después de la integración, se deben pasar al sistema todas las pruebas de unidad.
Ritmo sostenible	No se consideran aceptables grandes cantidades de horas extras, ya que a menudo el efecto que tienen es que reduce la calidad del código y la productividad a medio plazo.
Cliente presente	Debe estar disponible al equipo de la XP un representante de los usuarios finales del sistema (el cliente) a tiempo completo. En un proceso de la programación extrema, el cliente es miembro del equipo de desarrollo y es responsable de formular al equipo los requerimientos del sistema para su implementación.

Fuente: SOMMERVILLE, Ian. Ingeniería de *software*. p. 365.

En un proceso de XP, los clientes están fuertemente implicados en la especificación y establecimiento de prioridades de los requerimientos del sistema. Los requerimientos no se especifican como una lista de funciones del sistema. Más bien, los clientes del sistema son parte del equipo de desarrollo y discuten escenarios con otros miembros del equipo.

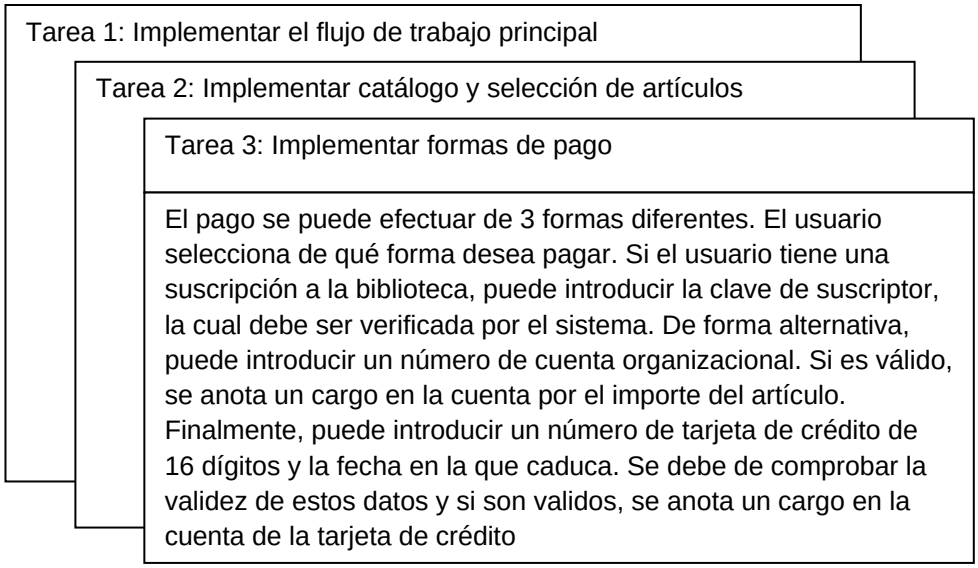
En la figura 3 se muestra un ejemplo de las tarjetas de historias que se tienen que realizar en XP.

Figura 3. **Tarjeta de historia para la descarga de documentos**

Descarga e impresión de un articulo
En primer lugar, seleccione el artículo que desea de una lista visualizada. Tiene entonces que decirle al sistema como lo pagara –se puede hacer a través de una suscripción, una cuenta de empresa o mediante una tarjeta de crédito. Después de esto, obtiene un formulario de derechos de autor del sistema Para que lo rellene. Cuando lo haya enviado, se descarga el articulo en su computadora Elija una impresora y se imprimirá una copia del artículo. Le dice al sistema que la impresión se ha realizado correctamente. Si es un artículo de solo impresión, no puede guardar la versión en PDF, Por lo que automáticamente se elimina de su computadora.

Fuente: SOMMERVILLE, Ian. Ingeniería de *software*. p. 366.

Figura 4. **Tarjetas de tareas para la descarga de documentos**



Fuente: SOMMERVILLE, Ian. Ingeniería de *software*. p. 367.

1.3.3. Roles de XP

En otras fuentes de información aparecen variaciones y extensiones de roles de XP, a continuación en la tabla IV se describen los roles de acuerdo a la propuesta original de *Beck*.

Tabla VI. **Roles de XP**

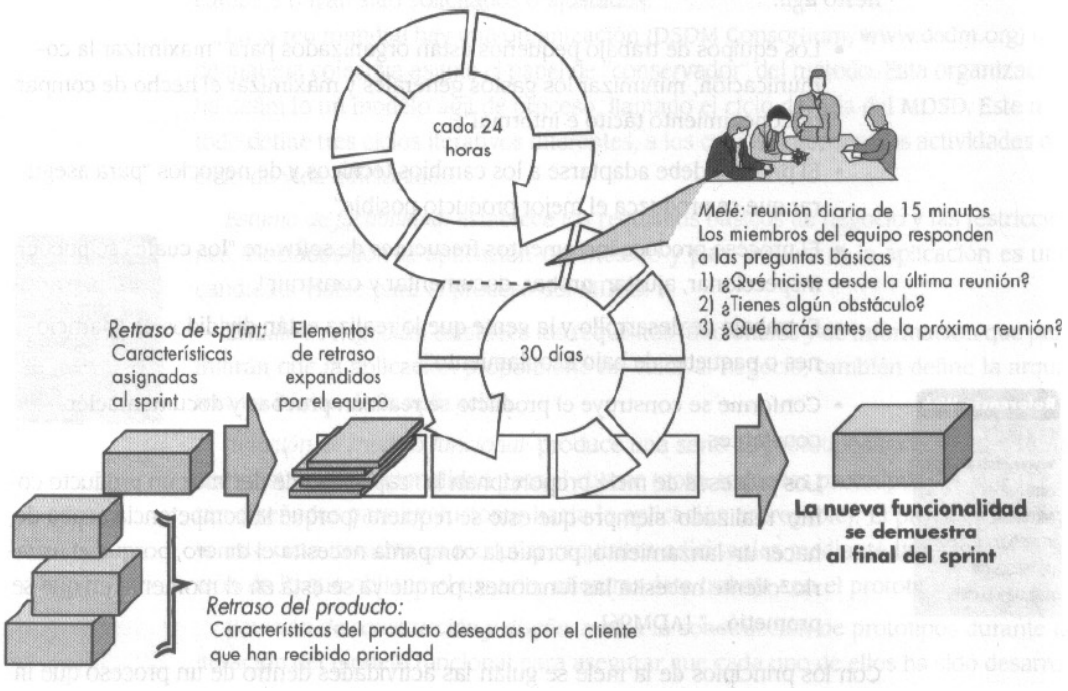
Rol	Descripción
Programador	El programador escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.
Cliente	El cliente escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
Encargado de pruebas(*Tester*)	El encargado de pruebas ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
Encargado de seguimiento (*Tracker*)	El encargado de seguimiento proporciona realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones. También realiza el seguimiento del progreso de cada iteración y evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes.
Entrenador (*Coach*)	Es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se continúe el proceso correctamente.
Consultor	Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto. Guía al equipo para resolver un problema específico.
Gestor (*Big boss*)	Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

Fuente: LETELIER, Patricia; PENADÉZ, M[a] Carmen. Metodologías ágiles para el desarrollo de *software*: *extreme programing* (XP). p. 10.

1.4. *SCRUM*

"*Scrum* fue desarrollado por *Jefff Sutherland* y su equipo a principios de la década de los 90, *scrum* también es conocido como *melé*, término derivado de una jugada de *rugby (*es un deporte de contacto en equipo, nacido en Inglaterra). En años recientes *Schwaber* y *Beedle* han presentado el desarrollo posterior de los métodos de *scrum*".[16]

Figura 5. **Flujo de proceso de *scrum***



Fuente: PRESSMAN, Roger. Ingeniería del *software* un enfoque práctico. p. 94.

[16] PRESSMAN, Roger. Ingeniería del *Software* un enfoque práctico (McGraw-Hill Interamericana), pág. 92.

"*Scrum* es un modelo de referencia que define un conjunto de prácticas y roles y se toma como punto de partida para definir el proceso de desarrollo que se ejecuta durante un proyecto".[17]

Scrum subraya el uso de un conjunto de "patrones de proceso de *software*" que ha probado su efectividad en proyectos con tiempos de entrega muy reducidos, requisitos cambiantes y condiciones críticas en los negocios.

1.4.1. Actividades de desarrollo de *scrum*

Cada uno de los patrones de proceso define un conjunto de actividades de desarrollo:

- "Retrasos: son una lista que considera la prioridad de los requisitos o características de proyecto que proporcionan un valor comercial para el cliente. El gerente de producto evalúa los retrasos y actualiza las prioridades de acuerdo con lo requerido.

- *Sprint*: consiste en unidades de trabajo que se requieren para satisfacer un requisito definido en los retrasos en un período predefinido (el lapso usual es de 30 días). Durante el *sprint* no se introducen cambios.

- Reuniones de *scrum*: son reuniones cortas (por lo general de 15 minutos) y las realiza a diario el equipo de *scrum*. Existen tres preguntas que plantean y responden todos los miembros del equipo.

  ¿Qué hiciste desde la última reunión?
  ¿Cuáles obstáculos encontraste?

[17] Wikipedia. "*Scrum*", 10 de septiembre 2010, http://es.wikipedia.org/wiki/Scrum (13 de septiembre de 2010).

¿Qué esperas lograr para la siguiente reunión del equipo?

Las reuniones las dirige el maestro de *scrum* y evalúa las respuestas de cada persona. Cada reunión de *scrum* ayuda al equipo a descubrir problemas potenciales tan pronto como sea posible.

- Demostración: se entrega el incremento de *software* al cliente de forma que, éste demuestre y evalué la funcionalidad implementada. Es importante señalar que la demostración quizá no contenga toda la funcionalidad planeada, sino aquellas funcionalidades susceptibles de entregarse dentro del período establecido".[18]

1.4.2. *Backlog*

Un proceso *scrum* reconoce tres *backlog*:

- "*Backlog* de producto: es un repositorio de requerimientos enunciados por los interesados en el éxito del proyecto (los llamados *stakeholders*, que son usuarios, administradores, técnicos de soporte, etc.).

- *Backlog* de versión: consiste en una lista de requerimientos extraídos del *backlog* de producto, priorizada para la próxima versión del producto. Los elementos tienen un mayor nivel de detalle en cuanto a los requerimientos y tienen asociadas estimaciones más precisas, realizadas por los miembros del equipo *scrum*.

[18] PRESSMAN, Roger. Ingeniería del *Software* un enfoque práctico (McGraw-Hill Interamericana), pág. 95.

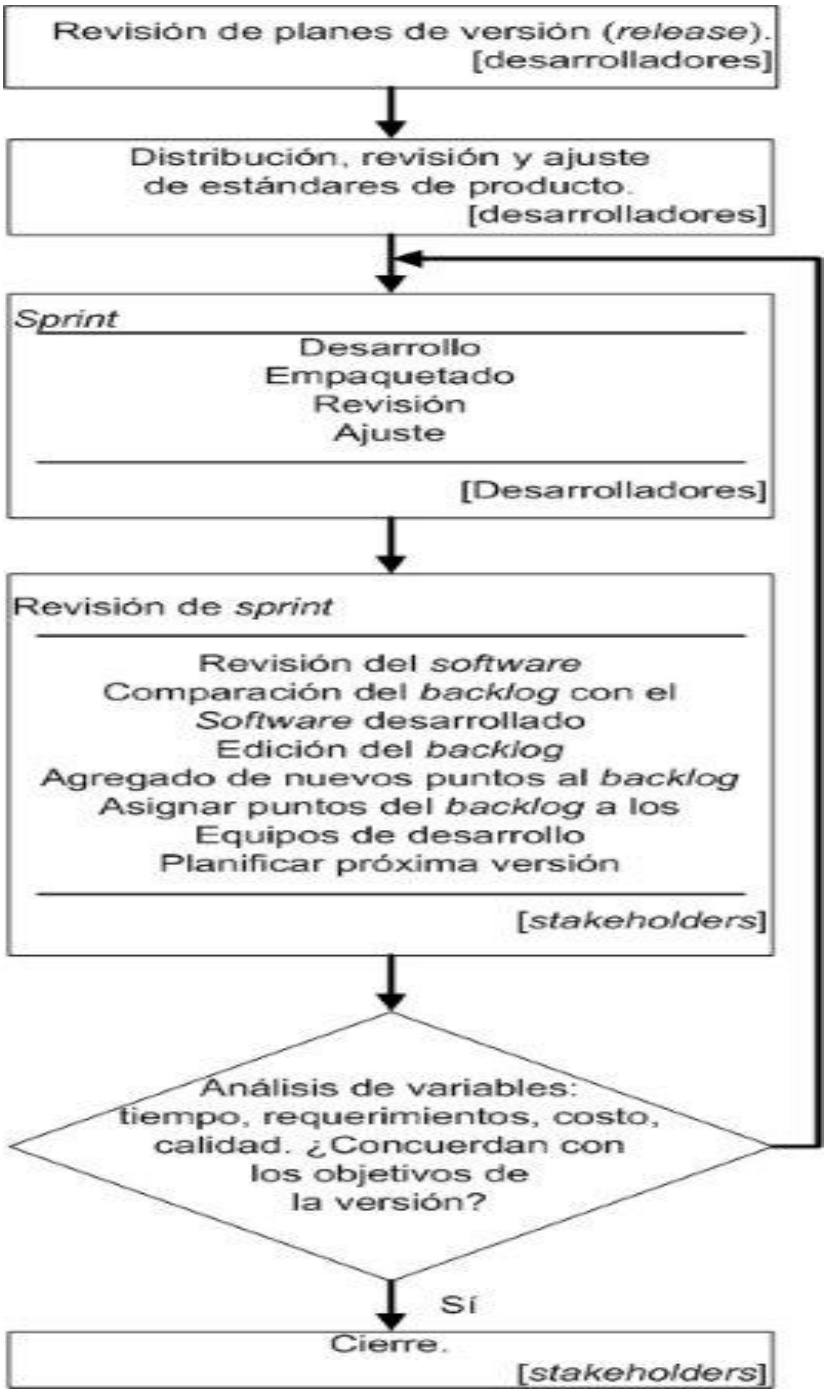
- *Backlog* de *sprint*: se arma el principio de cada *sprint* y reúne aquellos requerimientos que el equipo se compromete a completar para cuando finalice dicho *sprint*. Completar un requerimiento implica codificarlo, testearlo y documentarlo".[19]

1.4.3. Fases de *Scrum*

El proceso de desarrollo de *scrum* se compone de cinco actividades principales: revisión de planes de *reléase*; distribución, revisión y ajustes de los estándares de producto; *sprint*; revisión de *sprint*, y cierre, las cuales se muestran en detalle en la figura 6.

[19] DU MORTIER, Gustavo. "El Método *Scrum*", 19 de marzo de 2007, http://www.mastersoft.com.ar/MsWeb/otros_archivos/NotaScrumPCUsers.pdf (13 de septiembre de 2010).

Figura 6. **Fases de *scrum***



Fuente: DU MORTIER, Gustavo. El método *scrum*. p. 55.

1.4.4. Roles de *scrum*

Tabla VII. **Roles de *scrum***

Rol	Descripción
Dueño del producto	Las responsabilidades incluyen las características del producto, determinar la fecha de lanzamiento y el contenido, asegurar la rentabilidad del producto, priorizar las características según el valor del mercado, ajustar las características y prioridades.
Scrum master	Su responsabilidad es asegurar que el equipo se mantenga plenamente funcional y productivo, permite la cooperación estrecha entre todos los roles y funciones, elimina las barreras que obstaculicen el desarrollo del proyecto, protege al equipo de las interferencias externas y asegura que el proceso se lleve a cabo correctamente.
Equipo	Debe de ser poli funcional compuesto por siete miembros. Su labor consiste en seleccionar el objetivo final de cada *sprint*, especificar los resultados del trabajo y llevarlo a cabo.

Fuente: DU MORTIER, Gustavo. El método *scrum*. p. 57.

2. FUNCIONAMIENTO Y MANTENIMIENTO

2.1. Reglas de negocio

Describen las políticas, normas, operaciones, definiciones y restricciones presentes en una organización, que son de vital importancia para alcanzar los objetivos misionales. Las organizaciones funcionan siguiendo múltiples reglas del negocio, explícitas en procesos, aplicaciones informáticas, documentos, etc. Las reglas del negocio son dinámicas y están sujetas a cambios en el tiempo.

Cuando se desarrolla un *software* se tiene claro cuáles son las reglas del negocio. Ellas muestran el funcionamiento principal e indican las restricciones y limitantes que se tienen en la empresa, como por ejemplo: ayudan a definir que limitantes de *software*, base de datos que se puede utilizar, sobre que arquitectura se debe de desarrollar la aplicación (cliente-servidor, n-capas, etc.).

2.1.1. Herramientas para la gestión de reglas de negocio

Existen diferentes herramientas que permiten el manejo de las reglas del negocio, entre las cuales se pueden mencionar:

- *onRules (Delta-R)*
- *ILOGJrules*
- *RuleBurst*
- *Oracle Rules Engine*
- *Microsoft Business Rules Engine*
- *JBoss Rules*

- *Dinno Corp*
- *REM*
- *BizTalk Server*

En la tabla VIII se determina el estado actual de los departamentos de TI en la Municipalidad de Guatemala, en base al factor de las reglas del negocio. Que problemas han tenido por no tener el conocimiento de las reglas del negocio y los beneficios que se obtienen si se utiliza el factor en la implementación de nuevos sistemas.

Tabla VIII. **Reglas del negocio en la Municipalidad de Guatemala**

Departamento de TI	**Informática Municipal**	**Emetra**	**Empagua**	**Licencias de la construcción**	**Catastro**
Factor reglas del negocio					
Estado actual	No cuenta con documentación, relaciona a las reglas del negocio, el *software* es realizado en base a las reglas que el departamento imponga.	No cuenta con la documentación relacionada a las reglas del negocio, el *software* es realizado en base a las reglas que impone la persona que desarrolla el *software*.	Tienen el conocimiento de las reglas del negocio, pero no se encuentra documentado.	No cuenta con la documentación de las reglas del negocio.	No cuentan con la documentación relacionada a las reglas del negocio, esto, ocasiona que se realice un *software* que no cumple con las necesidades.
Problemas del estado actual	El departamento coloca las reglas del negocio en base a la experiencia que ha tenido.	La persona que no conoce las reglas del negocio desarrolla un *software* que no cumple con las necesidades de Emetra.	Esto puede ocasionar problema cuando exista un cambio de personal.	Desarrollar *software* que no cumple con las necesidades de Licencias de la construcción.	Seguir desarrollando *software* que no cumple con las necesidades de Catastro.
Beneficios de implementación	El *software* no es rechazado por los usuarios porque se realiza en base a las reglas del negocio.	El personal conoce y comprende cómo tiene que desarrollar el *software*.	Ayuda al nuevo personal a tener conocimiento de las reglas del negocio de Empagua.	Desarrollar *software* que cumple con las sus necesidades.	El conocimiento de las reglas del negocio evita desarrollar *software* que no cumple con las necesidades de Catastro.

Fuente: elaboración propia.

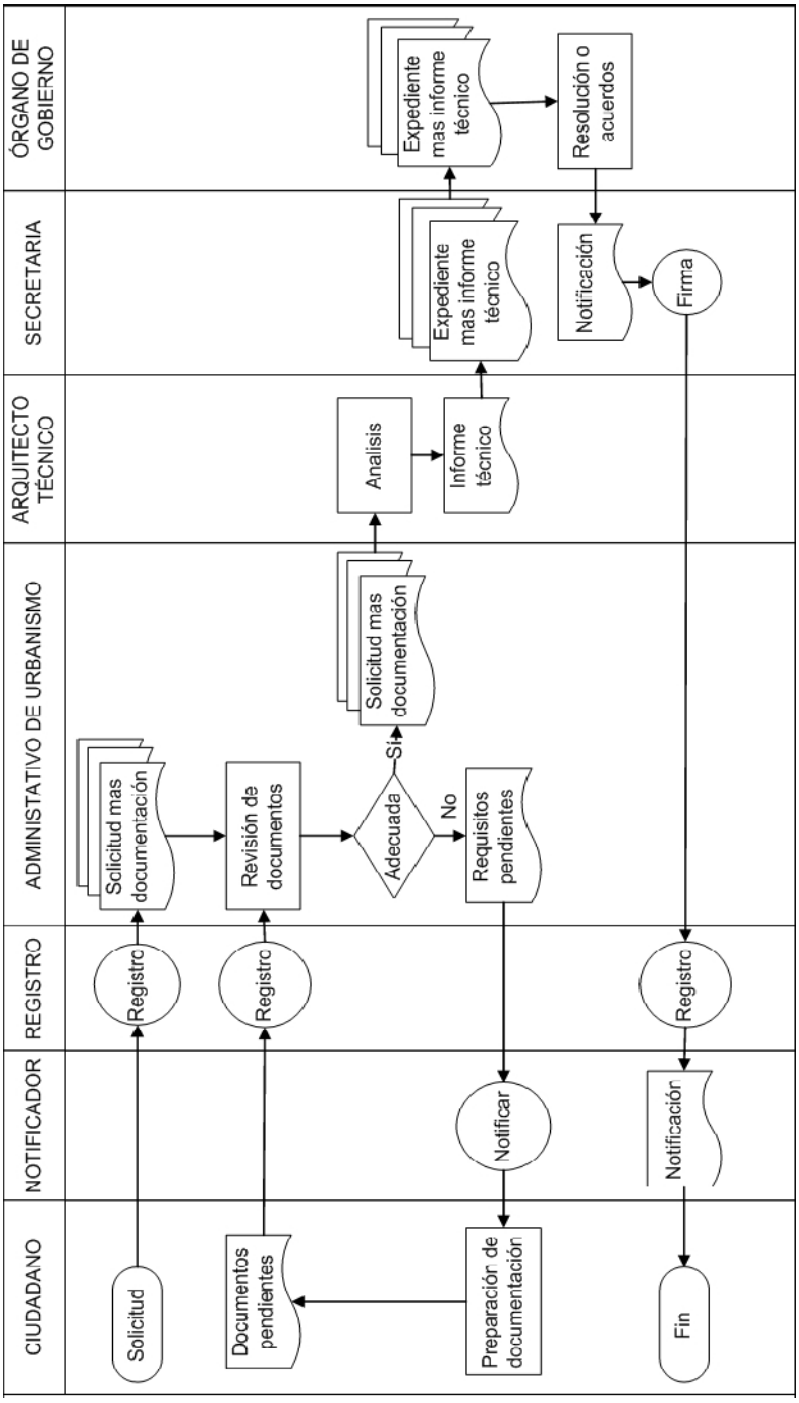
Las personas relacionadas en el desarrollo de un sistema tienen que tener amplio conocimiento sobre las reglas del negocio, si el personal no conoce el negocio, nos lleva al caso de Catastro (ver tabla VIII). Este factor es importante en el desarrollo de un sistema, porque garantiza que se realiza en base a los lineamientos de la empresa, un ejemplo es el *software* que la empresa utiliza. Existen muchas de las empresas que solicitan que el sistema tiene que realizarse en .Net con base de datos *SQL Server*, otras que sea en ambiente *web* utilizando *Personal Home Page* (PHP) con base de datos *Oracle*, las empresas realizan acuerdos con licencias de *software* y esto, se convierte en una regla del negocio.

2.2. Metodologías para el modelado de procesos y actividades

El modelado de procesos se utiliza para capturar y filtrar requerimientos, para documentar visualmente las funcionalidades de los agentes y permitir la comunicación con los expertos. Se utiliza para comprender las necesidades del cliente mediante diagramas, ayuda al personal encargado de la realización del proyecto y a los clientes porque visualizan de manera gráfica lo que ellos requieren y si realmente es lo que están solicitando.

El modelado de proceso y actividades muestra una mejor visión del sistema de *software* que se desarrolla. También permite apreciar la relación que existe entre las distintas actividades, analizar cada actividad, definir la relación que existe entre procesos y la identificación de los subprocesos que puedan existir.

Figura 7. **Ejemplo de modelado de procesos y actividades**



Fuente: aiteco consultores. Modelado de procesos. http://www.aiteco.com/modelpro.htm. Fecha de consulta: 27 de abril 2010.

En la figura anterior se muestra un ejemplo de cómo se puede realizar el modelado de procesos y actividades, éste debe de tener todos los procesos y actividades que se tienen que realizar para llegar al resultado deseado.

En la tabla IX se encuentra el estado actual de los departamentos de TI de la Municipalidad de Guatemala; problemas que se tienen por no utilizar el factor de modelado de procesos y actividades y los beneficios que pueden tener al utilizar el factor.

Tabla IX. **Modelado de procesos y actividades en la Municipalidad de Guatemala**

Departamento de TI	Informática Municipal	Emetra	Empagua	Licencias de la construcción	Catastro
Factor modelado de procesos y actividades					
Estado actual	Utiliza una metodología para el desarrollo de procesos y actividades.	No realiza un modelado de procesos y actividades, el *software* se realiza lo que se platica en reuniones.	Utiliza una metodología para el desarrollo de proceso y actividades.	Utiliza una metodología para el desarrollo de procesos y actividades.	El *software* se realiza en base a las solicitudes de los usuarios y no existe un flujo de procesos y actividades.
Problemas del estado actual	No existe problema en la realización de procesos y actividades.	Esto ha provocado que los usuarios no acepten el *software* y que existan demasiados cambios.	No existe problema en la realización de procesos y actividades.	El personal que realiza el modelado de procesos y actividades lo realiza en base al conocimiento y experiencia adquiridos, esto ha provocado que muchas veces se tenga que rediseñar.	Provoca que los usuarios realicen demasiados cambios al *software*.
Beneficios de implementación	Actualmente la metodología que se utiliza satisface las necesidades.	Ayuda a tener una mejor visión del sistema y permite que los usuarios puedan tener el conocimiento de cómo será el flujo del nuevo sistema.	Actualmente la metodología utilizada cumple con sus necesidades.	Si el personal tuviera los conocimientos técnicos y experiencias, de cómo realizar un modelado de procesos no fuera necesario realizar tantos cambios.	El *software* no necesita demasiados cambios porque los usuarios tienen el conocimiento del funcionamiento del mismo.

Fuente: elaboración propia.

Cuando se realiza un sistema sin utilizar un modelado de procesos y actividades, no se puede presentar al usuario un diagrama donde visualiza el funcionamiento del sistema. El diagrama se utiliza para que el usuario verifique y valide el flujo de los procesos y actividades e indicando si satisface sus necesidades, en este punto el usuario puede realizar cambios que no afectan en tiempo.

Lo que hacen los departamentos que no cuentan con un modelado de procesos es realizar el *software* y cuando ya se tiene un alto porcentaje de desarrollo se muestra al usuario, para que valide si funciona de acuerdo a sus necesidades. Esta metodología es inadecuada, porque si el usuario no está conforme con el *software*, realiza un cambio significativo que puede ocasionar que se desarrolle por completo una parte del sistema, esto involucra tiempo, costo y en la mayoría de los casos atraso en el tiempo de entrega.

Realizar un modelado de procesos, es un factor importante en el desarrollo de un sistema de *software*. Muestra la visión general del sistema, ayuda a los usuarios a observar de una manera entendible el funcionamiento del sistema, permite a los desarrolladores entender el flujo que debe de llevar el sistema.

2.3. Contratos

Un contrato es un acuerdo de voluntades, oral o escrito, manifestado en común entre sujetos (partes del contrato), que se obligan en virtud del mismo, regulando sus relaciones relativas a una determinada finalidad o cosa, cuyo cumplimiento debe ser de manera recíproca.

Es un acuerdo de voluntades que generan derecho y obligaciones sólo para las partes contratantes y sus causahabientes. Para realizar un contrato se establece cuáles serán los derechos y obligaciones que ambas partes tendrán, debido a que esto es una parte muy importante al momento de que una de ellas tenga el incumplimiento del contrato. Si el contrato se encuentra con partes ambiguas, puede causar que se tienda a confusión y problemas entre las partes involucradas.

2.3.1. Tipos de contratos

Existen diferentes formas de realizar un contrato, entre los cuales se encuentra:

- *Hardware*
- *Software*
- Confidencialidad
- *Outsourcing*

En la tabla X se encuentra una comparación de los departamentos que utilizan los contratos y qué beneficios obtienen por utilizar este factor. También se encuentran los departamentos que no utilizan los contratos y como pueden beneficiarse en la implementación de este factor.

Tabla X. **Contratos en la Municipalidad de Guatemala**

Departamento de TI	**Informática Municipal**	**Emetra**	**Empagua**	**Licencias de la construcción**	**Catastro**
Factor contratos					
Estado actual	Utiliza contratos de *hardware* y *software*.	No utiliza contratos.	Utiliza contratos de *software*.	No utiliza contratos.	No utiliza contratos.
Problemas del estado actual	No existe problema.	El personal que se encuentra en el departamento no puede cubrir con las necesidades de Emetra.	No existe problema.	No existe problema.	No existe problema.
Beneficios de implementación	Se tiene soporte del *software* y *hardware*, si existe problema en el *hardware* la empresa contratada tiene la responsabilidad de solucionar el problema.	Un contrato de *outsourcing* puede ayudar a mejorar el servicio en el departamento de TI de Emetra.	La empresa contratada realiza los cambios necesarios al *software* y corrige los errores que puedan suceder en el sistema.	El estado actual del departamento cumple con las necesidades de los usuarios.	El personal que se encuentra en el departamento cumple con las necesidades.

Fuente: elaboración propia.

Cuando se realiza un contrato de *software* se debe de verificar que las partes involucradas tengan sus responsabilidades y revisando que la información no contenga ambigüedades.

3. INICIO

3.1. Toma de requerimientos

"Es el proceso de descubrir, analizar, documentar y verificar los servicios y restricciones del sistema, que terminan siendo las bases del contrato entre el cliente y el desarrollador del *software"*.[20]

Un requerimiento es una parte de un sistema, el cual puede representar una capacidad, características o un factor de calidad de un sistema. La unión de todos los requerimientos son los que forman el sistema terminado. Para que un sistema cumpla con las necesidades del cliente, éste debe de cumplir con todos los requerimientos que él solicitó.

Existen diferentes formas de realizar la toma de requerimientos, entre los cuales se pueden mencionar:

- Encuestas cerradas: los clientes responden una encuesta rígida y cerrada.
- Encuesta abierta: los *stakeholders* son interrogados por el equipo desarrollador y así tener una mejor comprensión de los que el cliente necesita.
- Casos de uso: muestra la interacción que existe entre los actores y el sistema.

[20]SOMMERVILL, Ian. Ingeniería de *software*. p. 108.

3.1.1. Tipos de requerimientos

Existen básicamente 2 tipos de requerimientos los cuales son:

- Requerimientos funcionales: es lo que un sistema tiene que hacer en base a las necesidades de los clientes.
- Requerimientos no funcionales: son los requerimientos que no son solicitados por el cliente pero si son necesarios para que un sistema funcione de una mejor manera.

3.1.2. Características de los requerimientos

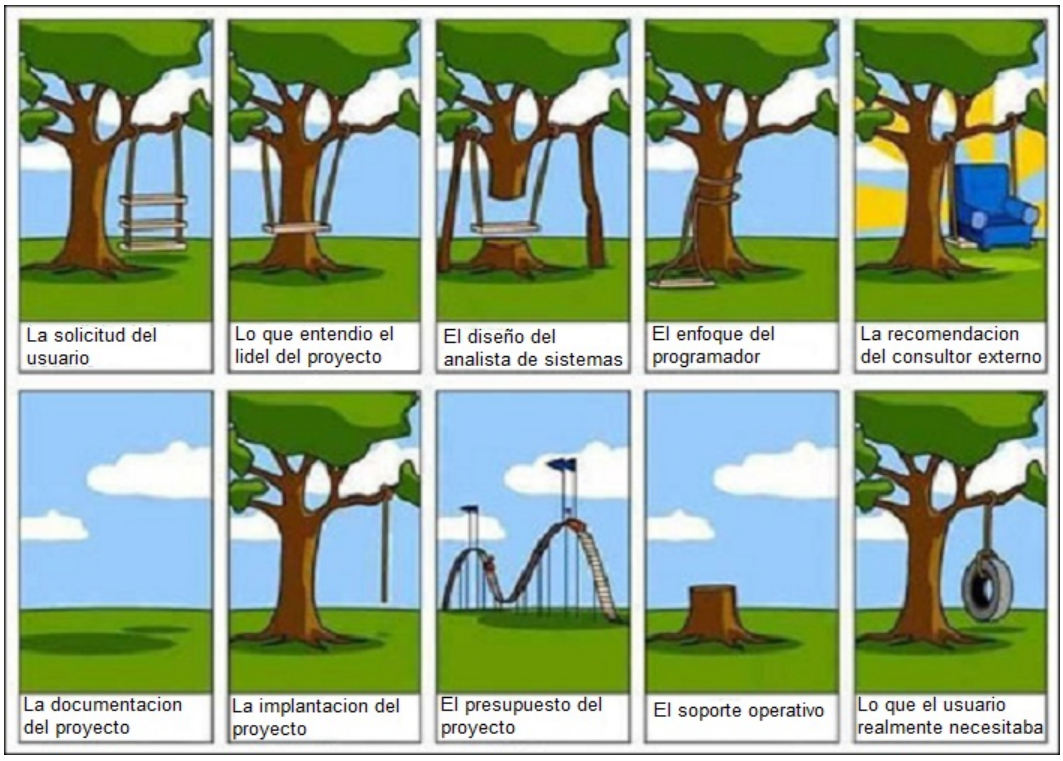
Los requerimientos bien formulados deben de satisfacer varias características, si no cumplen con éstas, deben de ser reformulados hasta hacerlo, las características son las siguientes:

- Necesario: debe ser necesario para el sistema.
- No ambiguo: el requerimiento debe ser claro, preciso y tener una única interpretación.
- Conciso: debe redactarse en un lenguaje entendible para los clientes.
- Consistente: ningún requerimiento tiene que entrar en conflicto con otro.
- Completo: los requerimientos deben contener toda la información necesaria, no se tiene que referenciar a fuentes externas.
- Alcanzable: tiene que ser realista y posible alcanzable con el dinero, tiempo y los recursos disponibles.
- Verificable: se tiene que poder validar con certeza y verificar si el requerimiento fue alcanzado o no.

El problema que enfrentan las empresas en la toma de requerimientos es, que no se consideran las características que deben de cumplir y cuando se desarrolla el sistema y se presentan al usuario las funcionalidades, se dan cuenta que no cumple con lo solicitado debido a que el requerimiento tiene partes ambiguas y presenta diferentes formas de interpretación.

A continuación se muestra una imagen sobre lo que usualmente ocurre cuando se toman los requerimientos, cada una de las personas involucradas lo entiende de diferente manera y al final se realiza un sistema completamente diferente a lo que el usuario necesita.

Figura 8. **Problema de interpretación en la toma de requerimientos**



Fuente: QUISPE OTAZU, Rodolfo. ¿Qué es la ingeniería de requerimientos?. http://www.rodolfoquispe.org/blog/que-es-la-ingenieria-de-requerimientos.php. Fecha de consulta: 19 de junio 2010.

En la tabla XI se determina el estado actual de los departamentos de TI de la Municipalidad de Guatemala que problemas han surgido por una mala implementación de los requerimientos y como pueden solucionar los problemas existentes.

Tabla XI. **Toma de requerimientos en la Municipalidad de Guatemala**

Departamento de TI	**Informática Municipal**	**Emetra**	**Empagua**	**Licencias de la construcción**	**Catastro**
Factor toma de requerimientos					
Estado actual	Utiliza la metodología RUP para la toma de requerimientos.	Utiliza reuniones con las personas involucradas.	Utiliza entrevistas con las personas involucradas para obtener los requerimientos.	Se utilizan reuniones con el personal para obtener los requerimientos.	Realiza reuniones con el personal involucrado.
Problemas del estado actual	En algunos casos la documentación de los requerimientos no es mostrada al usuario para que valide la información.	No se realiza documentación sobre los requerimientos.	No se presenta la documentación a los usuarios para validar si es lo que ellos necesitan.	No se presenta la documentación a los usuarios.	No se realiza documentación de los requerimientos.
Beneficios de implementación	Es conveniente que los usuarios revisen y validen la documentación de los requerimientos esto ayuda a minimizar los cambios del usuario.	Es necesario realizar la documentación correspondiente a los requerimientos para que los usuarios verifiquen y acepten el *software*.	La documentación se tiene que presentar al usuario para obtener sus puntos de vista y validaciones.	Los usuarios tienen que validar y verificar los requerimientos para evitar inconvenientes en el desarrollo del sistema.	La documentación de los requerimientos es necesaria para la validación y aceptación de los usuarios.

Fuente: elaboración propia.

Los requerimientos son los que definen la estructura del sistema, se deben de priorizar, validar y verificar que cumplan con las características. Si se realiza esto, garantizamos que el *software* realice lo que el usuario necesita, aumentando la probabilidad de éxito del *software*. Es importante que las empresas calendaricen el tiempo necesario para realizar la toma de requerimientos y enseñen a los desarrolladores a darle la importancia que tienen los requerimientos.

3.2. Usuarios y roles

Existen diferentes roles y usuarios en la creación de un nuevo sistema de *software*, un usuario es una persona que se encuentra involucrada directa o indirectamente con la realización de un proyecto; un rol es un papel que juega uno o más personas en el proyecto; los roles son las responsabilidades que obtienen las personas.

Existen diferentes categorías de usuarios en un proyecto, éstos pueden ser: clientes, usuarios finales, profesionales, gestores del proyecto y gestores superiores. Los clientes y usuarios finales son importantes en el desarrollo de un sistema de *software*, debido a que ellos son los que proporcionan los requerimientos del proyecto y validan la funcionalidad del *software*; los gestores superiores definen los aspectos de negocio; los gestores del proyecto son los que planifican, organizan y controlan a los que realizan el proyecto y los profesionales proporcionan las capacidades técnicas necesarias para la ingeniería de un producto.

Los roles genéricos son: analistas, desarrolladores, probadores y administradores. Los usuarios siempre desempeñan un rol en la realización de cualquier proyecto de desarrollo de *software*.

Tabla XII. **Usuarios y roles en la Municipalidad de Guatemala**

Departamento de TI	**Informática Municipal**	**Emetra**	**Empagua**	**Licencias de la construcción**	**Catastro**
Factor de usuarios y roles					
Estado actual	Utiliza los usuarios y roles del RUP.	No tiene usuarios y roles.	Utiliza usuarios y roles definidos por el departamento, no utiliza una metodología.	El departamento les asigna las responsabilidades y realizan cambios de personal durante el desarrollo del sistema.	No utilizan usuarios y roles en el desarrollo de un sistema.
Problemas del estado actual	No existe problema.	Los usuarios involucrados en el proyecto no tienen responsabilidades asignadas.	No existe problema.	Una nueva persona no sabe que tiene que realizar debido a que no existe una definición de los roles y usuarios.	Los usuarios no tienen responsabilidades en el desarrollo de un sistema.
Beneficios de implementación	La metodología que se utiliza actualmente satisface las necesidades.	La definición de los usuarios y roles permite la asignación de responsabilidades y evita el desorden.	La asignación de los usuarios utilizada satisface las necesidades de Empagua.	La utilización de los usuarios y roles satisface las necesidades pero el cambio de personal se debe de evitar.	Los usuarios involucrados tienen sus responsabilidades y todos saben que tiene que realizar durante el desarrollo del *software*.

Fuente: elaboración propia.

En la tabla anterior se encuentran definidos los departamentos de tecnología informática TI que sí utilizan usuarios y roles, los que no lo utilizan, que beneficios se pueden obtener al utilizarlos y problemas que surgen si no se utilizan estos.

En el desarrollo de un sistema es importante tener definidos los usuarios y roles, ayuda a la asignación de responsabilidades y tareas. Cuando no se manejan los usuarios y roles proporciona el desorden en el desarrollo e impacta en tiempo, cambios y en el peor caso en el desfase del tiempo de entrega del sistema. Los usuarios no comprenden la justificación del tiempo, ellos quieren el sistema en la fecha programa no importando los cambios que se realicen, por eso es necesario desde el inicio asignar las responsabilidades y que se comprenda la importancia de la asignación de los usuarios y roles en el desarrollo del sistema.

3.3. Requerimientos de *hardware*

Son las características necesarias que se tienen para obtener el mejor desempeño de las aplicaciones que se desarrollan. El *hardware* es una parte indispensable al momento de realizar un sistema de *software* debido a que si no se cuenta con las características necesarias, afecta en el rendimiento del mismo.

Cuando se desarrolla un *software* se debe de tomar en cuenta los requerimientos mínimos y recomendados de *hardware*, con esto se garantiza que no existe problema al momento de la instalación e implementación del sistema.

Al seleccionar el *hardware* se toma en cuenta también los requerimientos que realiza el cliente, debido a que con ellos es que determina la decisión sobre qué *hardware* se acopla más a las necesidades y si ya se cuenta con el mismo. Se realiza un análisis sobre el *hardware* que se tiene versus las necesidades de nuestros clientes y se valida si cumple con los requerimientos mínimos necesarios, si no cumple se debe de notificar al cliente e indicarle los problemas que pueden surgir si se implementa el *software* sobre el *hardware* existente. En la mayoría de estos casos se deja al cliente que tome la decisión, pero lo recomendable es contar con el *hardware* necesario.

Tabla XIII. **Requerimientos de *hardware* en la Municipalidad de Guatemala**

Departamento de TI / **Factor de *hardware***	**Informática Municipal**	**Emetra**	**Empagua**	**Licencias de la construcción**	**Catastro**
Estado actual	El *hardware* utilizado para el ambiente de producción es proporcionado por *International Business Machines* IBM.	Utiliza el mismo *hardware* que informática municipal.	El *hardware* es propio.	Utiliza el mismo *hardware* que informática municipal.	Utiliza el mismo *hardware* que informática municipal.
Problemas del estado actual	Cuando se desarrolla un nuevo sistema no se toma en cuenta los requerimientos del *hardware* existente, esto ha provocado problemas de rendimiento y en algunos casos quejas de los usuarios.	Cuando se desarrolla un nuevo sistema no se toma en cuenta los requerimientos del *hardware* existente.	No tienen problema, el *hardware* adquirido fue en base a los requerimientos del nuevo sistema que se realizó.	Cuando se desarrolla un nuevo sistema no se toma en cuenta los requerimientos del *hardware* existente y realizan un nuevo sistema sin verificar si tienen el *hardware* adecuado.	Cuando se desarrolla un nuevo sistema no se toma en cuenta los requerimientos del *hardware* existente.
Beneficios de implementación	La utilización de los requerimientos del *hardware* evita problemas de rendimiento, escalabilidad, etc.	La utilización de los requerimientos del *hardware* evita problemas de rendimiento, escalabilidad, etc.	En la realización de nuevos sistemas tienen que tomando en cuenta las especificaciones del *hardware*.	La utilización de los requerimientos del *hardware* evita problemas de rendimiento, escalabilidad, etc.	La utilización de los requerimientos del *hardware* evita problemas de rendimiento, escalabilidad, etc.

Fuente: elaboración propia.

En la tabla anterior se encuentran los problemas que tiene actualmente la Municipalidad de Guatemala por no tomar en cuenta los requerimientos de *hardware* cuando se realiza un nuevo sistema. Este problema se puede evitar cuando se realiza un análisis de los requerimientos necesarios del *hardware* para el *software* que se desea desarrollar.

El *hardware* es una de las partes importantes en la implementación de un sistema y en muchas empresas no le dan importancia a este detalle, al final se dan cuenta que tienen un problema, que se hubiera evitado si se realiza el análisis correspondiente del *hardware*.

3.4. Requerimientos de *software*

Son las características de *software* necesarias para desarrollar un sistema, ejemplo: base de datos, lenguaje de programación, herramientas para pruebas, etc. Cuando se desarrolla un nuevo sistema se tiene que considerar las necesidades del cliente y con esto se selecciona el *software* que se utiliza durante el desarrollo.

Cuando se selecciona el *software* se debe tomar en cuenta: licenciamiento, ventajas, desventajas, compatibilidad de *hardware*, etc. Los requerimientos de *software* van de la mano con los requerimientos de *hardware*, debido a que no se utiliza un *software* si no se cuenta con el *hardware* necesario para el funcionamiento adecuado.

También existen los requerimientos que solicita el cliente para el *software* que se está desarrollando, estos requerimientos son esenciales en el desarrollo del nuevo sistema, debido a que de ello depende la aceptación o rechazo del cliente. Un ejemplo de estos requerimientos, es que el usuario desea tener acceso al *software* desde cualquier lugar, incluyendo el teléfono y otros dispositivos móviles, entonces en el desarrollo del *software* se tiene que tomar en cuenta esta solicitud del cliente.

Tabla XIV. **Requerimientos de *software* en la Municipalidad de Guatemala**

Departamento de TI	**Informática Municipal**	**Emetra**	**Empagua**	**Licencias de la construcción**	**Catastro**
Factor de *software*					
Estado actual	Utiliza *developer* 6i para las aplicaciones cliente servidor; *php* y apache para las aplicaciones *web*; la base de datos que utiliza es *oracle* 10g; actualmente existe la implementación de *Systeme*, *Anwendungen und Produkte* SAP.	Utiliza *developer* 6i para la realización de sus aplicaciones; la base de datos que utiliza es la misma que informática municipal.	Utiliza *developer* 6i , *java y aspx* para el desarrollo de las aplicaciones y la base de datos es *oracle* 10g.	Utiliza *developer* 6i, *developer* 10g para la realización de las aplicaciones, la base de datos que utiliza es la misma que informática municipal.	utiliza *developer* 6i para las aplicaciones cliente servidor, la base de datos es *oracle* 10g y *shape file* para la *geodatabase*.
Problemas del estado actual	No existe problema.	No existe problema.	No existe problema.	No existe problema.	No existe problema.
Beneficios de implementación	La utilización actual del *software* satisface las necesidades de los usuarios.	La utilización actual del *software* satisface las necesidades de los usuarios.	La utilización actual del *software* satisface las necesidades de los usuarios.	La utilización actual del *software* satisface las necesidades de los usuarios.	La utilización actual del *software* satisface las necesidades de los usuarios.

Fuente: elaboración propia.

En la tabla anterior se presenta el estado actual de los departamentos de tecnología informática TI de la Municipalidad de Guatemala y si tienen algún problema en la utilización del *software* actual.

3.5. Definición de estándares

"Un estándar es un conjunto de criterios documentados para especificar la adecuación de una acción u objeto. El administrador del proyecto es responsable de especificar los estándares de rendimiento esperado".[21] Los estándares en un proyecto de *software* juegan un papel importante debido a que ellos ayudan a desarrollar de mejor manera un sistema.

Existen diferentes tipos de estándares en el desarrollo de un sistema de *software*, pero los importantes son: documentación, código, pruebas, *hardware*, *software*, calidad. Las empresas que se dedican al desarrollo de *software* se les ayuda en el manejo de estándares debido a que, el personal cambia constantemente y con esto logran que se pueda integrar de una manera fácil a una nueva persona.

En la tabla XV, los departamentos que no utilizan un estándar definido tienen el problema de que cuando una persona que desarrolla el sistema no se encuentra y es necesario realizar un cambio urgente, entonces un compañero tiene que verificar el código y realizar cambio, esto se dificulta debido a que no entiende la estructura del código, por lo tanto se lleva más tiempo y afecta a la Municipalidad.

[21] VEGA FERNÁNDEZ, Juan Antonio. "estándares en la ingeniería del software", 1999, http://www.rogeliodavila.com/tcs/TCS%20Notes%20JAVega/Parte_06_OrgStd.ppt (21 de agosto 2010).

Tabla XV. **Definición de estándares en la Municipalidad de Guatemala**

Departamento de TI	Informática Municipal	Emetra	Empagua	Licencias de la construcción	Catastro
Factor de estándares					
Estado actual	Utiliza un documento donde se identifican los estándares de desarrollo.	No existe documentación para la definición de estándares; el desarrollador realiza el *software* utilizando sus propios estándares.	No tiene definido los estándares, los que son definidos por el personal que desarrolla el *software*.	La definición de estándares las realiza el personal que desarrolla el *software*.	No existe una definición de estándares.
Problemas del estado actual	No se tiene estándar definido para la estructura de la base de datos.	Cuando existe cambio de personal es difícil realizar mantenimiento al *software*.	Si existe un cambio de personal, dificulta realizar el mantenimiento del *software*.	Cuando existe cambio de personal es difícil realizar mantenimiento al *software*.	La realización de cambios necesita más tiempo debido a que no existe un estándar definido.
Beneficios de implementación	Si existe un cambio de administrador de la base de datos, la nueva persona puede entender de mejor manera la estructura de la base de datos.	La definición de todos los estándares necesarios para el desarrollo de un sistema ayuda a los programadores a entender el código desarrollado por otras personas.	La definición de todos los estándares necesarios para el desarrollo de un sistema, ayuda a los programadores a entender el código desarrollado por otras personas.	La definición de los estándares permite que el nuevo personal pueda entender en menos tiempo el código de otra persona.	Con los estándares el mantenimiento se realiza en menos tiempo.

Fuente: elaboración propia.

4. DESARROLLO

4.1. Tipos de pruebas

Para garantizar el perfecto funcionamiento de un sistema de *software*, se deben realizar ciertas pruebas para validar que el sistema cumpla con los requerimientos propuestos inicialmente y con su función principal, brindar a los usuarios un *software* eficiente y sin fallos o retrasos. Es importante mencionar que el rendimiento del sistema está estrictamente ligado con el *hardware*, se recomienda realizar las pruebas con el *hardware* que se utilizará para el sistema. El error que se comete cuando se realizan las pruebas es, utilizar un *hardware* diferente al que se usará con el sistema, esto provoca que los resultados obtenidos no tengan la credibilidad esperada.

Otra razón por la cual es necesario realizar las pruebas, es para comprobar si existe un error o un retraso importante que pueda afectar el funcionamiento del sistema, debido a que se requiere que sea rápido, estable y confiable. Además, si llegara a ocurrir un error en el sistema, se le debe informar al usuario la situación del mismo, por ejemplo: si existe un error de conexión con la base de datos, se le debe hacer saber al usuario mostrándole un mensaje amigable con el error.

Existen diferentes formas de realizar las pruebas, las dos más importantes o esenciales que se deben realizar en un sistema de *software* son:

- Pruebas de *stress*
- Pruebas unitarias

4.2. Pruebas de *stress*

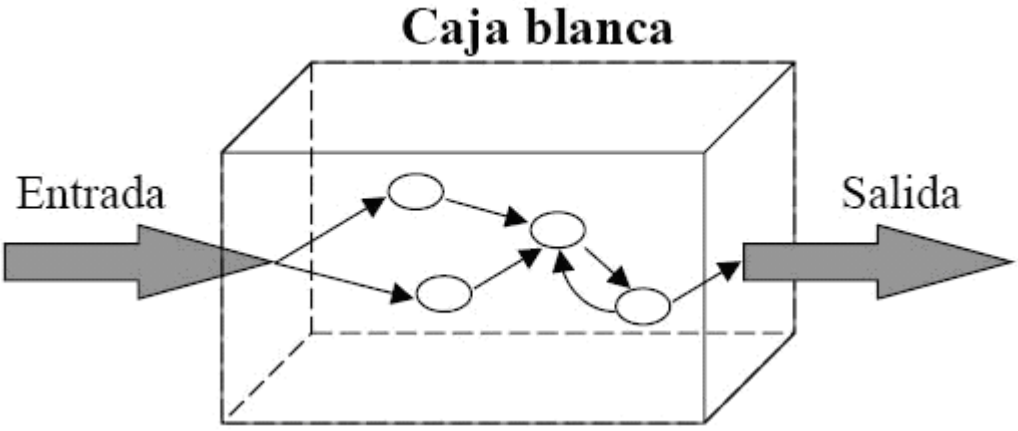
Estas se realizan para medir la capacidad máxima que puede soportar el sistema y las condiciones en las cuales trabaja realizando una cantidad definida de peticiones y procesos, ayuda a los administradores del sistema a validar si la aplicación rendirá lo suficiente en caso de que la carga real supere a la carga que se espera.

4.3. Pruebas de unitarias

Cuando se desarrolla un sistema las primeras pruebas que se deben tomar en cuenta son las pruebas unitarias, también son conocidas como pruebas de caja blanca o pruebas modulares, debido a que permiten verificar si un módulo funciona correctamente, estas pruebas no son las mismas que realiza el programador o desarrollador, por lo cual no se deben confundir.

Cuando se tiene que realizar las pruebas de un módulo, se calcula el tamaño del mismo para verificar si éste permite realizar las pruebas de manera sencilla y rápida, también es importante separar los módulos según su funcionalidad. Si un módulo del sistema es demasiado grande y contienen muchas funcionalidades se volverá más complejo de probar y al encontrar algún error se vuelve más difícil encontrar la funcionalidad defectuosa y corregirla.

Figura 9. **Esquema de pruebas unitarias**



Fuente: ORÉ B., Alexander. *Unit testing* – pruebas unitarias – cap 1.

En la figura anterior se observa cuál es el esquema que se sigue cuando se realizan pruebas unitarias, lo primero es tener datos de entrada, luego el sistema realiza las operaciones correspondientes dando como resultado las salidas, los datos de salida son los que se deben de analizar para validar el módulo y realizar los cambios necesarios si fuera el caso.

4.3.1. Características de las pruebas unitarias

Para que las pruebas unitarias sean buenas deben de cumplir con las siguientes características:

- Automatizable: no se requiere una intervención manual
- Completas: debe de cumplir la mayor cantidad de código
- Repetibles o reutilizables: no se deben de crear pruebas que sólo se puedan ejecutar una vez
- Independientes: la ejecución de una prueba no debe afectar a otra

- Profesionales: las pruebas se deben de considerar igual que el código

4.3.2. Ventajas

El objetivo que buscan las pruebas unitarias es, separar cada parte del sistema y mostrar que las partes individuales funcionen correctamente, las cuales proporcionan cinco ventajas:

- Fomenta el cambio: facilitan que el programador cambie el código para mejorar la estructura
- Simplifican la integración: permite llegar a un grado alto de seguridad de que el código está funcionando correctamente
- Documenta el código: las pruebas son la documentación, debido a que ahí se puede visualizar como utilizarlo
- Separación de la interfaz y la implementación
- Los errores están más acotados y son más fáciles de localizar

4.3.3. Desventajas

- Las pruebas unitarias no descubren todos los errores del código
- Sólo prueban las unidades por separado, por lo tanto no se descubren los errores de integración, problemas de rendimiento
- Sólo son efectivas si se utilizan en conjunto con los otros tipos de pruebas

4.4. Herramientas para tipos de pruebas

Existen diversidades de herramientas para realizar las pruebas, la herramienta se tiene que seleccionar según el tipo de prueba que se desea realizar. Podemos utilizar herramientas que tengan un tipo de licenciamiento y otras que son gratis, las que necesitan licenciamiento es necesario pagar para poder utilizarlas y las gratis se pueden descargar desde cualquier sitio en internet sin costo alguno.

La selección de la herramienta dependerá de que tipos de resultados que se desea obtener de las pruebas realizadas, dependerá también del tamaño del proyecto, por ejemplo: no se puede seleccionar una herramienta que soporte hasta 1.000 usuarios, si se desea realizar la prueba con un máximo de 50.000 usuarios, otro punto importante es el lenguaje donde se realizó el sistema de *software* y así pueden existir otros puntos importantes para la selección de la herramienta a utilizar.

4.4.1. Herramientas para pruebas de stress

Existe una gran variedad de herramientas para la realización de estas pruebas, entre las que se pueden mencionar tenemos:

Herramientas que necesitan licenciamiento

- *Open load tester*: se puede utilizar en sistemas *web*, si desea verificar el funcionamiento de esta aplicación la pueden descargar del siguiente sito: http://www.opendemand.com/openload/

- *Team edition for software testers*: utilidad para realizar pruebas de *visual studio .net*. Se puede descargar la versión de prueba del siguiente sitio: http://www.microsoft.com/downloads/en/details.aspx?familyid=572E1E71-AE6B-4F92-960D-544CABE62162&displaylang=en

Herramientas *opensource*

- *JMeter*: realizar pruebas en aplicaciones *web*, se puede descargar del siguiente sitio: http://jakarta.apache.org/jmeter/
- *Microsoft application center test*: se puede descargar del siguiente sitio: http://www.csharphelp.com/2007/06/microsoft-application-center-test/

4.4.2. Herramientas para pruebas unitarias

Entre las herramientas que existen para realizar estas pruebas se pueden mencionar:

Herramientas *opensource*

- *JUnit*: entorno de pruebas que existe para *java*. Se puede descargar del sitio: http://www.junit.org/home
- *TestNG*: es una herramienta complementaria para *JUnit*. Se puede descargar del sitio: http://testng.org/doc/index.html
- *SimpleTest*: herramienta para utilizarla en *PHP*. Se puede descargar del sitio: http://www.simpletest.org/en/download.html
- *PHPUnit*: *framework* para realizar pruebas en *PHP*. Se puede descargar del siguiente sitio: http://sourceforge.net/projects/phpunit/
- *CCPUnit*: *framework* para pruebas en *C/C++*. Se puede descargar del siguiente sitio: http://sourceforge.net/projects/cppunit/

- *NUnit*: *framework* para pruebas en *.net*. Se puede descargar del siguiente sitio: http://www.nunit.org/index.php?p=download

Tabla XVI. **Definición de estándares en la Municipalidad de Guatemala**

Departamento de TI / Factor definición de estándares	Informática Municipal	Emetra	Empagua	Licencias de la construcción	Catastro
Estado actual	Realiza pruebas unitarias al *software*, las pruebas las realiza el programador; otra persona se encargada de certificar la aplicación; las pruebas que realiza el programador no tienen documentación. No se realizan pruebas de estrés.	Realiza pruebas unitarias, las pruebas son realizadas por el programador y no se realiza documentación. No se realizan pruebas de estrés.	Las pruebas unitarias las realiza el desarrollador y el usuario, no existe documentación. No se realizan pruebas de estrés.	Las pruebas las realiza el desarrollador y el usuario, no existe la documentación. No se realizan pruebas de estrés.	Las pruebas las realiza el desarrollador y el usuario, no existe documentación de las pruebas realizadas. No realizan pruebas de estrés.
Problemas del estado actual	Las pruebas que se realizan al *software* siempre son diferentes y si existe un cambio en el *software* no se realizan nuevamente las pruebas anteriores esto ha provocado que surjan errores cuando el usuario utiliza el sistema. La falta de pruebas de estrés en algunos sistemas ha provocado que se degrade el rendimiento debido a la carga de operaciones.	Las pruebas no quedan documentadas y en muchos casos el *software* es liberado con demasiados errores, esto ha provocado la insatisfacción de los usuarios.	Las pruebas no quedan documentadas y cuando los usuarios prueban el sistema encuentran demasiados errores, esto ha provocado desconfianza en el *software*.	Las pruebas no quedan documentadas y cuando los usuarios prueban el sistema encuentran demasiados errores esto ha provocado desconfianza en el *software*. La falta de pruebas de estrés en muchas ocasiones ha provocado que los usuarios no puedan realizar las operaciones	Las pruebas no quedan documentadas y cuando los usuarios prueban el *software* encuentran demasiados errores esto ha provocado desconfianza en el *software*.
Beneficios de implementación	Al realizar todos los tipos de pruebas necesarios al *software* se puede evitar el problema de rendimiento, integración y desconfianza de los usuarios, esto ayuda al departamento de TI.	Utilizando los tipos de pruebas se evitan los errores y los usuarios tendrían mayor confianza en el *software*.	Los usuarios confían en el *software* y la información de las pruebas realizadas se utiliza para volver a utilizarlas si existe un cambio en el *software*.	El rendimiento del *software* se obtiene mediante las pruebas de estrés y la confianza en el sistema se puede recuperar realizando pruebas unitarias para disminuir los errores.	Las pruebas realizadas se encuentran documentadas y si existen errores se corrigen, esto provoca que los usuarios tengan mayor confianza en el *software* que se desarrolle en catastro.

Fuente: elaboración propia.

En la tabla anterior se encuentra el estado actual de los departamentos de TI, como puede mejorar el desarrollo de software si se implementa el factor de estándares.

Los usuarios si observan que un sistema tiene demasiados errores cuando lo prueban, tiende a crearles el paradigma de que, al final el sistema no funcionara correctamente y que les ocasionará demasiados problemas, por eso es conveniente que exista el área de *quality assurance* QA para los sistemas que se desarrollen.

4.5. Trabajo en equipo

Es un factor primordial en el desarrollo de un sistema de *software*, debido a que se puede contar con todas las herramientas necesarias para el proyecto, el tiempo suficiente, pero si las personas que se encuentran dentro del proyecto no pueden realizar su trabajo en equipo, es seguro que el proyecto llegará al fracaso.

Cuando se realiza el trabajo en equipo, las actividades fluyen de manera rápida y eficiente, sin embargo no es fácil que los miembros de un mismo grupo se entiendan entre sí, debido a que cada uno piensa diferente.

Existen diferentes ventajas cuando se trabaja en equipo, entre ellas tenemos:

- Se trabaja con menos tensión
- La responsabilidad se comparte al buscar soluciones de diferentes puntos de vista
- Intercambio de opiniones respetando las ideas de los demás

- Se comparten los incentivos económicos y reconocimientos profesionales
- Se experimenta de forma positiva de un trabajo bien hecho

Cuando el trabajo en equipo fracasa, existen diferentes factores que influyen sobre ésto, entre los cuales se pueden mencionar:

- No existe un clima agradable de trabajo
- Se planifica incorrectamente
- Existe negatividad y egoísmo en el grupo
- Desmotivación de los miembros
- Los involucrados no se sienten parte del grupo
- No existe la confianza mutua
- Los objetivos a cumplir no están claros

4.5.1. Diferencia entre trabajo en equipo y grupo de trabajo

En la tabla siguiente se muestra una serie de características que existen entre trabajar en grupo o en equipo.

Tabla XVII. **Diferencia entre trabajo en equipo y grupo de trabajo**

Grupo de trabajo	Trabajo en equipo
Liderazgo fuerte e individualizado.	Liderazgo compartido.
Responsabilidad individual.	Responsabilidad individual y colectiva.
La formación de un grupo de trabajo ocurre a partir de su creación o instalación.	La formación de un equipo de trabajo es un proceso de desarrollo.
Enmarca su acción dentro del objetivo global de la organización.	Dentro del marco del objetivo global de la organización, se auto asignan propósitos y metas específicas.
Sus resultados son vistos como suma del esfuerzo individual.	Sus resultados se toman y evalúan como producto de un esfuerzo conjunto de sus miembros.
El trabajo colectivo se considera como algo inevitable o un mal necesario.	El trabajo colectivo se observa como una oportunidad y se disfruta.
Los conflictos se resuelven por imposición o evasión.	Los conflictos se resuelven por medio de confrontación productiva.
Se encuentra centrado principalmente en la tarea.	Se centra en la tarea y en el soporte socio - emocional de sus miembros.
No reconoce diferencias de valores, juicios e incompetencias entre sus miembros.	Se reconocen e incorporan las diferencias como una adquisición o capital del equipo.

Fuente: GÓMEZ MUJICA, Aleida. Acerca del trabajo en grupos o equipo. http://www.bvs.sld.cu/revistas/aci/vol11_6_03/aci10603.htm. Fecha de consulta: 16 de junio 2010.

El trabajo en equipo garantiza que se pueda terminar un *software* a tiempo y con la calidad deseada, esto es debido a que todos los miembros se encuentran sincronizados y cada uno realiza lo que tiene que hacer y como lo tiene que realizar. En la figura 10, se visualiza que trabajando en equipo podemos realizar una tarea grande sin necesidad de realizar mucho esfuerzo, en caso contrario si se trabaja solos, tenemos que dividir esta tarea y realizarla por segmentos, ésto significa más tiempo y por lo tanto más costo a las empresas.

Figura 10. **Ejemplo de trabajo individual y trabajo en equipo**

Fuente: monografías. Liderazgo y trabajo en equipo. http://www.monografias.com/trabajos26/liderazgo-y-equipo/liderazgo-y-equipo.shtml. Fecha de consulta: 26 de mayo 2010.

En la tabla XVIII, se explica los problemas que surgen en los departamentos de TI de la Municipalidad por no utilizar el trabajo en equipo y los beneficios que se pueden obtener al realizar el trabajo en equipo.

Tabla XVIII. **Trabajo en equipo, Municipalidad de Guatemala**

Departamento de TI	**Informática Municipal**	**Emetra**	**Empagua**	**Licencias de la construcción**	**Catastro**
Factor trabajo en equipo					
Estado actual	Se fomenta el trabajo en equipo pero no existe información de cómo se tiene que trabajar en equipo.	No se utiliza el trabajo en equipo.	Fomentan el trabajo en equipo para el desarrollo de *software*.	No se utiliza el trabajo en equipo.	Fomentan el trabajo en equipo para el desarrollo de *software*.
Problemas del estado actual	Todos saben que tienen que trabajar en equipo pero no se conoce que es el trabajo en equipo, este problema es necesario solucionarlo porque con solo saber que es necesario trabajar en equipo no significa que estemos trabajando en equipo.	No existe un problema de trabajo en equipo porque una persona realiza todo.	No tienen información relacionada al trabajo en equipo.	No tienen información relacionada al trabajo en equipo.	No tienen información relacionada al trabajo en equipo.
Beneficios de implementación	Ayuda a que las personas tengan una mejor comunicación y comprensión cuando desarrollan un nuevo sistema para la Municipalidad de Guatemala.	En Emetra se tiene una persona que realiza todo el desarrollo, análisis y diseño, por lo cual no necesitan el trabajo en equipo.	Ayuda a que las personas tengan una mejor comunicación y comprensión cuando desarrollan un nuevo sistema para Empagua.	Ayuda a que las personas tengan una mejor comunicación y comprensión cuando desarrollan un nuevo sistema para licencias de la construcción.	Ayuda a que las personas tengan una mejor comunicación y comprensión cuando desarrollan un nuevo sistema para catastro.

Fuente: elaboración propia.

La resistencia al cambio es una de las barreras que suelen impedir que las personas trabajen en equipo, debido a que muchas tienen un concepto equivocado sobre el trabajo en equipo, suelen pensar que si realizan este cambio perderán la importancia que tienen en la empresa.

5. APLICACIÓN

5.1. Documentación

Documentar un *software* no es sólo colocar diagramas, contratos, pruebas, etc., esto es parte de una documentación pero no es todo lo que se debe de colocar del *software*. El error que se comete muchas veces cuando se realiza un sistema es que, todo lo relacionado a documentación se deja como secundario, esto provoca que el sistema se realice en base a lo que, cada una de las personas involucradas hayan comprendido.

Un sistema sin documentación tiene un alto porcentaje del fracaso, desde la primera reunión hasta la entrega del sistema, debe de contener documentación. Por ejemplo: cuando se toman los requerimientos, éstos deben de estar documentados, las personas involucradas deben estar de acuerdo con lo que está escrito, debido a que esto es lo que dará vida al sistema, los documentos son un apoyo tanto para el personal que está realizando el sistema como para los clientes.

Una buena documentación tiene que estar clara, sencilla y debe contener lo que el cliente necesita conocer del sistema, ejemplo: debe saber cómo se inicia un proceso hasta como finaliza una actividad, como debe realizar una acción para obtener el resultado que desea.

Cuando se realiza la documentación de un *software*, es recomendable que éstos tengan una plantilla definida, existen diferentes plantillas que se pueden utilizar, pero si se desea se puede crear una plantilla propia, esto es con la finalidad de que no existan documentos con diferente tipo de letra, tamaño, estructura, etc., ayuda a que los usuarios comprendan de mejor manera lo que se está expresando.

5.2. Manual de usuario

Es uno de los documentos importantes, debido a que en él, se encuentra cómo el usuario tiene que interactuar con el sistema, la forma de uso del sistema. Un error que cometen las empresas que desarrollan *software* es que, no le ponen importancia al manual del usuario y no lo realizan de una manera adecuada.

Un punto importante que se debe tomar en cuenta es que, el manual se tiene que realizar cuando el sistema se encuentra en el 100% funcional y aprobado por los clientes. Otro de los errores comunes que se observa es que se realiza el manual de usuario aún cuando se está desarrollando el sistema y esto provoca que existan demasiados cambios al documento y muchas veces por error humano no se realizan todos los cambios necesarios y cuando los usuarios del *software* acuden al manual, observan que no se encuentra la información que necesitan para realizar un proceso o actividad.

5.2.1. Estructura de un manual de usuario

Cada empresa puede adoptar su propia estructura para el manual del usuario, la siguiente es una posible estructura que se puede utilizar cuando creamos un manual.

5.2.1.1. Prefacio

Debe de contener la información de cómo se puede utilizar el manual.

5.2.1.2. Índice

Debe mostrar la tabla del contenido del manual, para que el usuario pueda encontrar de una manera sencilla lo que desea consultar.

5.2.1.3. Guía rápida de cómo utilizar funciones principales del sistema

En esta parte se debe de colocar las funcionalidades más importantes y utilizadas por el usuario, por ejemplo: si el manual es sobre un sistema contable, debe de tener como crear una partida presupuestaria, pólizas, cierres contables, etc., aquí se debe colocar de forma clara y sencilla cómo se debe de interactuar con el sistema para obtener el resultado deseado.

5.2.1.4. Explicación funcionamiento

Debe tener toda la funcionalidad del sistema, incluyendo la parte de funciones principales, solo que aquí se detalla un poco más el punto anterior, para que el usuario tenga una mejor idea del funcionamiento total del sistema.

5.2.1.5. Sección solución de problemas

Esta parte es importante debido a que aquí tiene que colocarse como se puede solucionar un error o alerta que proporciona el sistema, debe de estar bien detallada y clara la solución posible o soluciones que se pueden dar.

5.2.1.6. Preguntas frecuentes

En esta sección se colocan lo que suele pasar frecuentemente en el sistema, por ejemplo: cómo puedo ingresar al sistema, que puedo hacer en el sistema, etc.

5.2.1.7. Glosario

Se debe colocar la definición y explicación de cada uno de los términos utilizados en el manual, esto ayuda al usuario cuando no tiene un conocimiento sobre un término que se encuentra en el manual, así como a tener una mejor comprensión de la parte del sistema que desea entender.

Tabla XIX. **Manual de usuario, Municipalidad de Guatemala**

Departamento de TI	Informática Municipal	Emetra	Empagua	Licencias de la construcción	Catastro
Factor manual de usuario					
Estado actual	Realizan manual de usuario para las todas las aplicaciones, existe una persona que realiza el manual.	La falta de personal provoca que el manual de usuario no sea realizado, esto provoca que los usuarios no tengan la documentación de cómo se tiene que utilizar el sistema.	Realizan manual de usuario para las aplicaciones.	Realizan manual de usuarios para las todas aplicaciones.	Realizan el manual de usuarios para las aplicaciones.
Problemas del estado actual	No existe problema.	Los usuarios tienen que llamar al programador que realizo el sistema para que les explique el funcionamiento del sistema.	No existe problema.	No existe problema.	No existe problema.
Beneficios de implementación	Los usuarios pueden resolver dudas que tengan relacionados al uso del sistema.	Los usuarios pueden resolver dudas que tengan relacionados al uso del sistema.	Los usuarios pueden resolver dudas que tengan relacionados al uso del sistema.	Los usuarios pueden resolver dudas que tengan relacionados al uso del sistema.	Los usuarios pueden resolver dudas que tengan relacionados al uso del sistema.

Fuente: elaboración propia.

El manual de usuario muchas veces no se realiza en la Municipalidad porque los usuarios no exigen el manual y el personal de los departamentos de TI observan que los usuarios usualmente no leen el manual o no le toman importancia, ésta ha sido una decisión errónea sobre no realizar el manual de usuario, porque si los usuarios actuales se retiran y se contrata a nuevo personal, no tendrán un documento donde puedan obtener la información sobre el funcionamiento del sistema.

La persona que realiza los manuales, debe tener experiencia en la realización de estos documentos, lo más importante es transmitir la información a los usuarios en el lenguaje común para que sea comprendido.

Si no es posible contratar a una persona para realizar los manuales, el desarrollador puede realizar el manual y presentarlo a un usuario para que lo verifique y presente las observaciones sobre el documento, realizar cambios si el usuario lo indica, esta es una opción aunque no se recomienda implementarla debido a que le quita tiempo a las personas que desarrollan el sistema.

5.3. Manual técnico

Este explica el funcionamiento del sistema en palabras entendibles para las personas que necesiten realizar mantenimientos y no conocen sobre el funcionamiento del mismo. Este manual debe de explicar qué componentes se utilizaron, configuraciones, interacción entre las partes del sistema, etc. Muchas personas cometen el error de realizar este manual explicando partes del código y no colocan información importante que sirve para comprender el sistema de forma técnica.

La información que se coloca en este manual no es igual a la información que se coloca en el manual de usuario, esto se debe a que, las personas que consultarán los manuales no son del mismo tipo. En el manual técnico las personas deben tener el conocimiento técnico de esta área, en el otro son personas que no necesariamente deben tener el conocimiento en el área.

5.3.1. Estructura de un manual técnico

No existe una plantilla definida para la realización de este manual, a continuación se explica una posible estructura de cómo se puede realizar un manual técnico.

5.3.1.1. Objetivo y alcances del sistema

En esta parte se deben colocar los objetivos que cumple el sistema y hasta donde llega el sistema, es decir, si el sistema solo funciona en un ambiente cliente servidor, *web*, etc. Aquí también se define qué es lo que el sistema realiza de la organización, como por ejemplo: sólo existe el módulo de contabilidad, compras, ventas y recursos humanos, los otros módulos no se realizaron en el sistema.

5.3.1.2. Manual de normas, políticas y procedimientos de la organización en las que se basa el sistema para su implementación

Se deben colocar las reglas del negocio, adicionalmente si existen limitantes en el uso de *software*, ejemplo: la empresa sólo utiliza *software* con licencia o *software opensource* o permite utilizar ambos, etc., adicionalmente existen empresas que deben cumplir con leyes y reglamentos del país.

En esta sección se debe colocar toda la información necesaria para que cuando se desea realizar cambios en el sistema se tomen en consideración y entiendan que tiene que seguir estos lineamientos.

5.3.1.3. Descripción de *hardware*

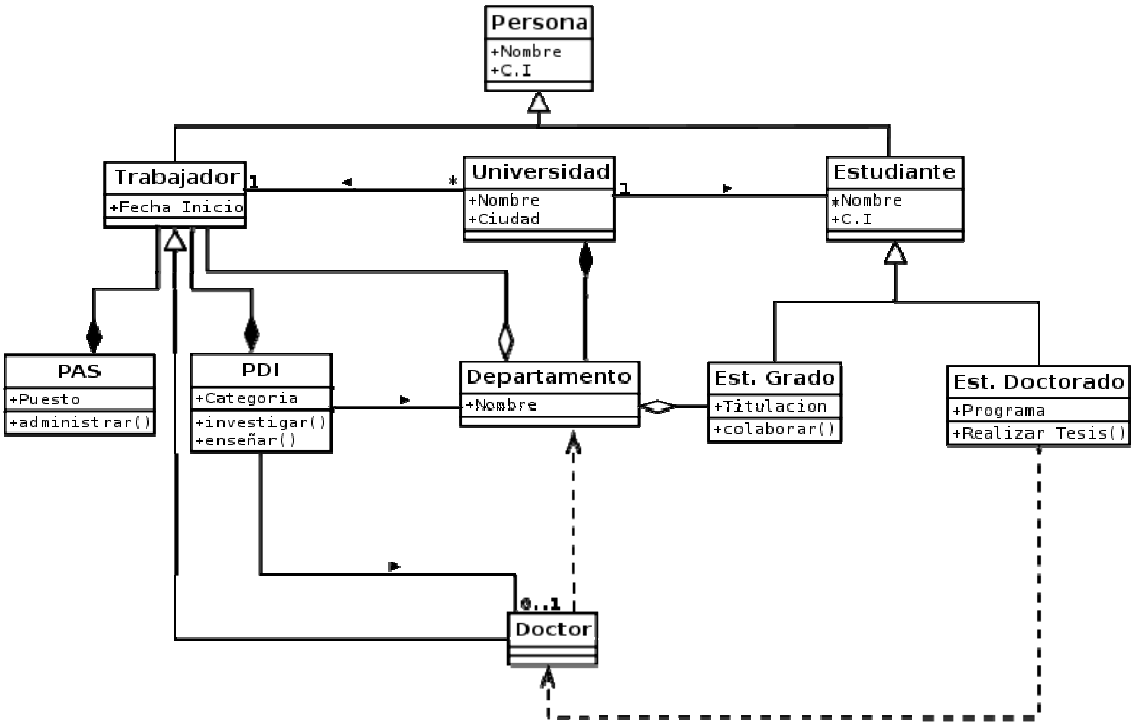
Es necesario colocar la información técnica del *hardware* para que el sistema funcione de manera óptima, es conveniente colocar los requerimientos mínimos y recomendables.

5.3.1.4. Diagrama de clases

Un diagrama de clases es un diagrama estático que describe la estructura de un sistema mostrando las clases, atributos y relaciones que existen entre ellas, este diagrama se realiza en el análisis y diseño de los sistemas, es importante colocarlo en este manual para que las personas tengan un panorama general sobre el sistema, en este diagrama se crea el diseño conceptual de la información que se maneja en el sistema y qué componentes se encarga del funcionamiento y la relación entre uno y otro.

A continuación en la Figura 11, se muestra un ejemplo sencillo de cómo se tiene que realizar un diagrama de clases.

Figura 11. **Ejemplo de diagrama de clases**



Fuente: wikipedia. Diagrama de clases. http://es.wikipedia.org/wiki/Diagrama_de_clases. Fecha de consulta: 18 de junio 2010.

5.3.1.5. Descomposición de módulos

En esta sección se tiene que colocar las partes que forman cada uno de los módulos del sistema, ésto ayuda a las personas a entender cada una de las piezas del módulo completo, cuando se realiza un cambio por personas que no realizaron el sistema, esta información es relevante porque pueden encontrar que parte del módulo debe ser cambiada para obtener el resultado deseado.

5.3.1.6. Descripción de proceso

Se tiene que colocar los procesos que se utilizan en el sistema, como por ejemplo: un proceso que calcula impuesto al valor agregado IVA, sueldo de los empleados, etc. Esto ayuda a las personas que no se involucraron en el desarrollo del sistema a entender que procesos se realizaron en el sistema y cuál se tiene que modificar si es necesario realizar un cambio, la descripción de procesos se debe realizar de forma técnica, como por ejemplo: cálculo del IVA, la fórmula que se utilizó.

5.3.1.7. Dependencia entre módulos

Esta sección es importante debido a que en ella se tiene que describir cómo se encuentra la relación entre los módulos, cómo uno depende de otro u otros para su correcto funcionamiento.

5.3.1.8. Dependencia entre procesos

Se coloca la dependencia que existe entre los procesos del sistema, ésto ayuda a las personas a validar el impacto que puede obtenerse si se realiza un cambio en un proceso.

5.3.1.9. Descripción de bases de datos

Se coloca el modelo Entidad-Relación para obtener una visión general de la estructura de la base de datos,

5.3.1.10. Diccionario de datos

Se definen las características lógicas de los datos que se van a utilizar en el sistema, este debe tener un nombre, descripción, alias y tipo, en esta sección tenemos que colocar cada uno de los campos que forman la base de datos y los datos que son importantes en el sistema, ejemplo: es el IVA, impuesto sobre la renta ISR, etc.

5.3.1.11. Diseño de reportes y pantallas

Se define una plantilla para la creación de reportes y pantallas, ésta ayuda a las personas a utilizar el mismo estándar que existe en el sistema cuando se tiene que desarrollar un nuevo reporte o pantalla.

Tabla XX. **Manual técnico, Municipalidad de Guatemala**

Departamento de TI	Informática Municipal	Emetra	Empagua	Licencias de la construcción	Catastro
Factor manual técnico					
Estado actual	No se realiza manual técnico pero existe documentación técnica de los sistemas que se desarrollan, es una metodología adquirida recientemente.	No existe manual técnico, la documentación técnica es solamente la estructura de la base de datos.	No existe manual técnico pero si utilizan documentación técnica para el desarrollo de *software*.	No existe manual técnico, realizan documentación técnica de los sistemas.	No existe manual técnico.
Problemas del estado actual	No existe problema debido a que la documentación técnica expresa la información necesaria de los nuevos sistema, pero de los sistemas anteriores la documentación se realiza por la persona que tiene el conocimiento del sistema, si no se encuentra una persona con el conocimiento del sistema no es posible realizar una buena documentación técnica.	Si se desea obtener información técnica de un sistema actual, no existe.	La documentación que realizan es suficiente para cumplir con las necesidades de Empagua.	La documentación realizada cumple con las necesidades de licencias de construcción.	Cuando se solicita información sobre los sistemas, el personal del departamento de TI realiza la documentación y muchas veces no contiene toda la información.
Beneficios de implementación	Se recomienda que validen si la documentación que realizan, cumple con las necesidades técnicas de la documentación.	Se puede obtener información técnica de los sistemas realizados, ayuda a entender el funcionamiento interno de los sistemas, como se deben de realizar los nuevos cambios y los módulos que afecta un cambio.	Es conveniente revisar periódicamente la documentación.	Se recomienda que validen si la documentación que realizan cumple con las necesidades técnicas de la documentación.	Realizar una documentación técnica desde que se desarrolla el sistema, es más completo porque el personal entiende de mejor forma el funcionamiento del *software*.

Fuente: elaboración propia.

En la figura anterior se muestra por qué el manual técnico es importante que lo realicen los departamentos de TI de la Municipalidad de Guatemala, este documento es necesario para que el nuevo personal pueda obtener la información técnica de los sistemas que existen, como es la dependencia entre módulos, interacción, etc.

5.4. Gestión de cambios

Cuando se desarrolla un sistema los cambios son inevitables, éstos se mantienen constantes en el desarrollo del proyecto, éstos pueden ser porque el cliente necesita un cambio o porque existe un error, la realización de los cambios de forma descontrolada provoca caos en el proyecto, retrasos en la entrega y problemas de calidad, por ejemplo: si no se lleva el control de los cambios, esto puede ocasionar que los desarrolladores distintos, efectúen modificaciones en un mismo punto del sistema y provoque problemas de calidad o que nuestro compañero de equipo tenga que rehacer parte del trabajo.

Cualquier empresa que se dedica al desarrollo de *software*, debe tener una metodología para manejar los cambios, personal que evalué y priorice las solicitudes, además es importante evaluar el impacto que tendrá en el proyecto, tanto en tiempo y costo, cuando las empresas no tiene una forma de manejar los cambios, ésto ocasiona que los proyectos fracasen porque no cumplen con el tiempo planeado y se salen del presupuesto.

Cuando se desarrolla un sistema, lo ideal es que no existan demasiados cambios, si esto llega a pasar, es conveniente realizar un análisis completo sobre cuál es el motivo que ocasiona los cambios, esto puede pasar si se realiza un mal levantado de requerimientos, los usuarios no proporcionan la información completa, esto último, es lo que usualmente sucede en los proyectos y para solucionarlo es conveniente tener un buen análisis y levantado de requerimientos, que los usuarios firmen y acepten los requerimientos que se están colocando en la documentación.

Si no se utiliza una metodología para desarrollar los cambios, las áreas del negocio y los usuarios suelen estar insatisfechos, por varias razones:

- No se sabe cuánto se tardarán en resolver el cambio al no existir ningún tipo de acuerdo ni compromiso.
- Los mismos cambios se repiten, y desde el departamento de TI no pueden evitarlo.
- Cada vez que llama un usuario se le hacen las mismas preguntas sobre sus datos de contacto, su ubicación, el *personal computer* PC que utiliza, etc.
- Durante el tiempo que dura el cambio no se le envía ningún tipo de comunicación sobre el avance de la resolución.
- En algunos casos los cambios se dan por cerrado sin que se haya comprobado si realmente se ha restablecido el servicio, lo que puede implicar volver a llamar para repetir nuevamente el mismo ciclo.
- Al final, siempre se opta por la “llamada a un conocido” del departamento de TI antes que por los cauces formales.

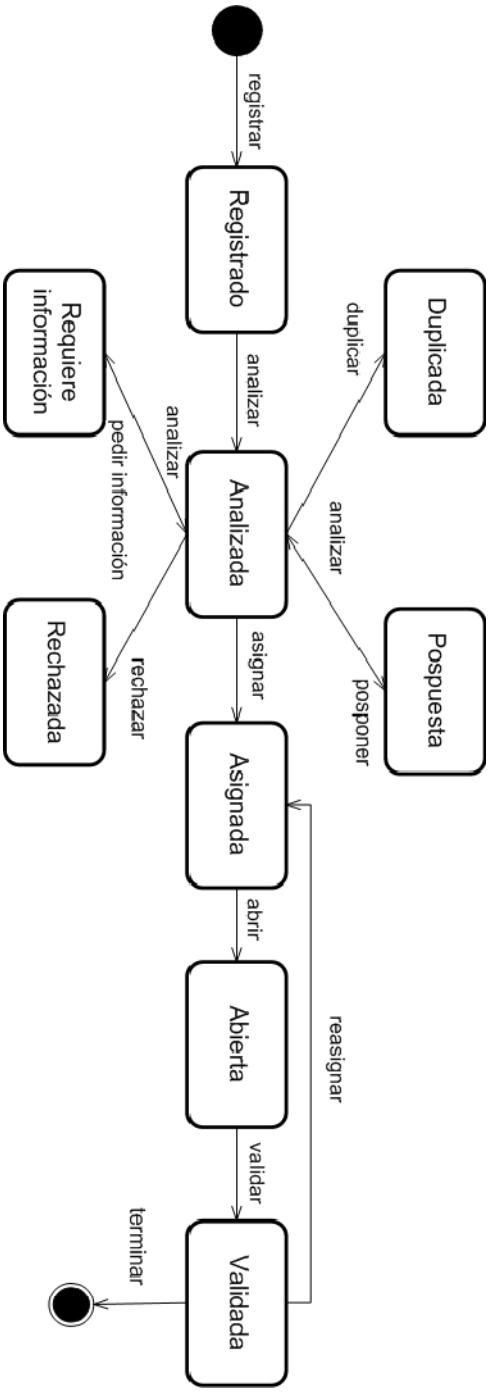
- La atención a clientes y usuarios constituye la imagen principal del área de TI, por lo que saber actuar con eficacia ante la resolución de cambios es un factor fundamental para no dañar aún más su imagen, que de por sí, no suele estar especialmente bien valorada.

5.4.1. Herramientas para automatizar la gestión de cambios

Existen varias herramientas que nos permiten administrar los cambios de *software*, estas herramientas pueden ser utilizadas con licencia u *opensource*, se habla un poco sobre IBM *Rational Clear Quest*, esta herramienta nos proporciona la capacidad de crear, actualizar, administrar y dar seguimiento a los cambios de acuerdo a las necesidades, en *Clear Quest* todo cambio debe de tener un ciclo de vida, que describe el flujo de posibles acciones y estados que puede seguir una solicitud.

En la figura 12 se muestra un ejemplo de un ciclo de vida para una solicitud de cambio que se puede configurar en *Clear Quest*, esta herramienta permite utilizar diferentes tipos de cambios y cada tipo de cambio debe de tener un ciclo de vida.

Figura 12. **Diagrama de proceso para la gestión de cambios**



Fuente: BAUTISTA, Ysaila; CALLERO, Rigoberto. Administración de cambios de software. http://www.sg.com.mx/content/view/271. Fecha de consulta: 23 de junio 2010.

Es importante mencionar que *Clear Quest* no es solamente para llevar el seguimiento de cambios, sino que permite la integración con otras herramientas de IBM que nos pueden ayudar y dar seguimiento a los proyectos de *software*.

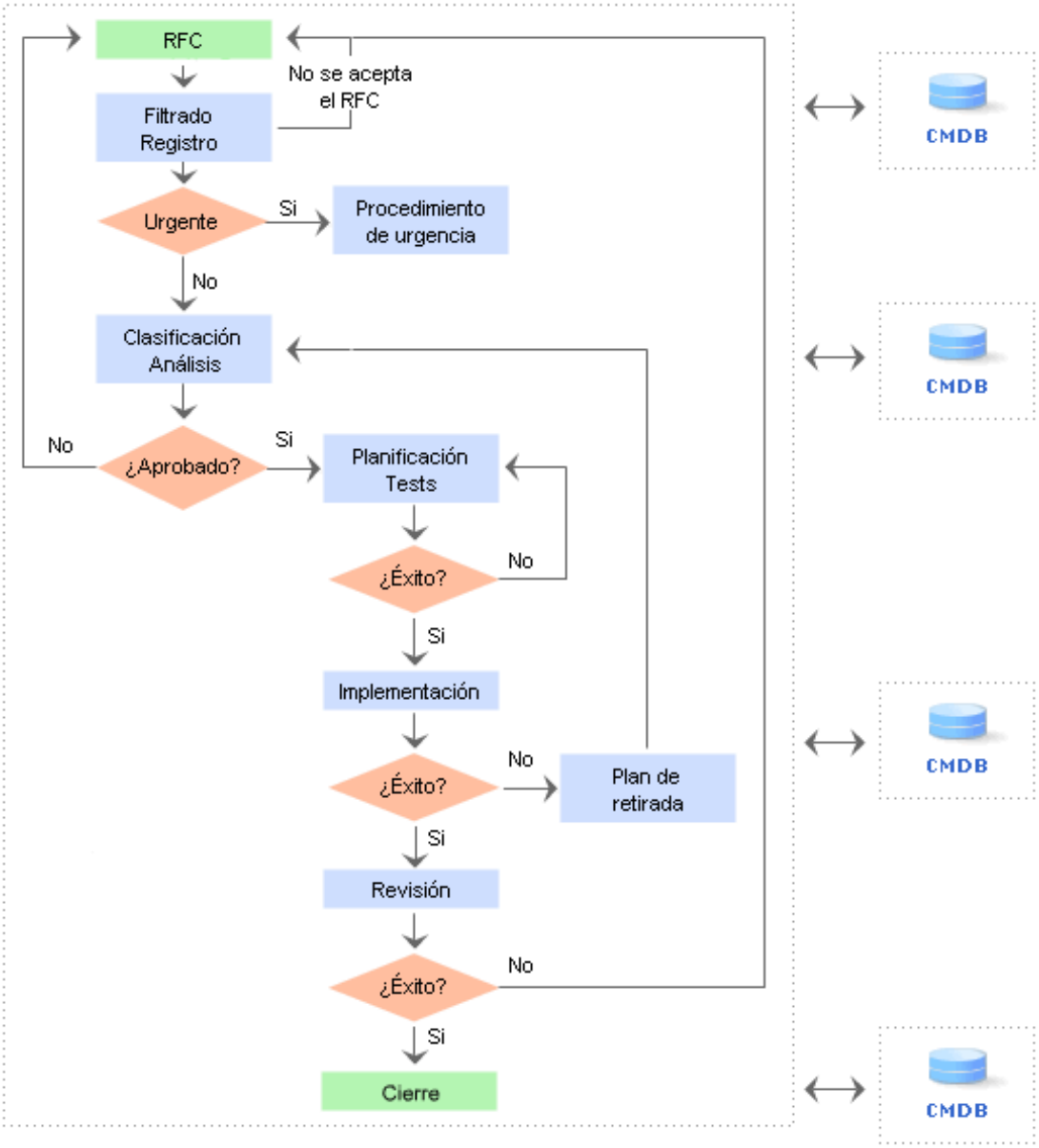
Para la gestión de cambios no es obligatorio tener una herramienta que nos permita llevar el control de los cambios, se puede llevar el control de los cambios utilizando un flujo de proceso o actividades desarrollado por la empresa, esto significa que cada empresa puede administrar los cambios de diferente manera, pero la finalidad es la misma, satisfacer a los clientes y tener un *software* de calidad.

5.4.2. Gestión de cambios utilizando *Information Technology Infrastructure Library* ITIL

Es un conjunto de buenas prácticas dirigidas a la gestión de servicios TI, es importante no generar falsas expectativas, ITIL por sí mismo no soluciona los incidentes, pero ofrece una guía para establecer el proceso adecuado para gestionarlos.

En la figura 13 podemos observar una guía de ITIL presentada por la organización Osiatis que se puede seguir cuando se presenta un cambio.

Figura 13. **Proceso para la gestión de cambios de ITIL**



Fuente: osiatis. Gestión de cambios. http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/gestion_de_cambios/introduccion_objetivos_gestion_de_cambios/introduccion_objetivos_gestion_de_cambios.php. Fecha de consulta: 26 de junio 2010.

Las principales actividades de la gestión de cambios son:

- “Monitorear y dirigir todo el proceso de cambio.
- Registrar, evaluar y aceptar o rechazar las *request for changes* RFCs recibidas.
- Convocar reuniones del *change advisory board* CAB, excepto en el caso de cambios menores, para la aprobación de las RFCs y la elaboración del calendario de cambios.
- Coordinar el desarrollo e implementación del cambio.
- Evaluar los resultados del cambio y proceder a su cierre en caso de éxito”.[22]

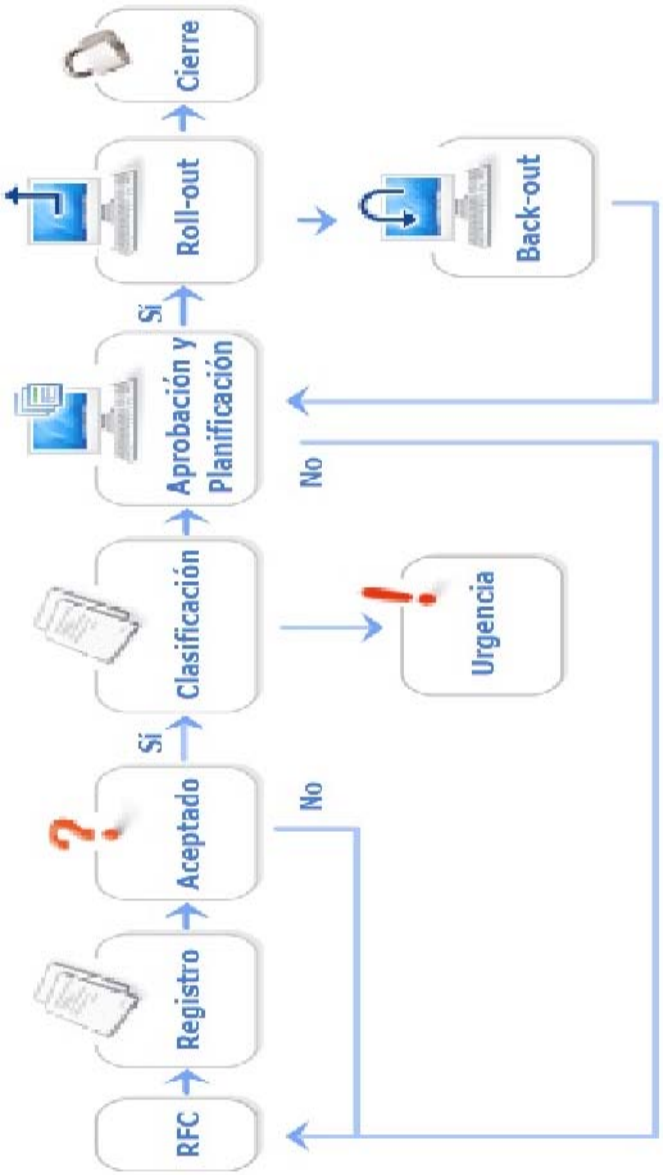
El proceso que presenta la organización Osiatis utilizando ITIL es el siguiente:

- Registro
- Aceptación y clasificación
- Aprobación y planificación
- Implementación
- Evaluación
- Emergencias

[22]Osiatis. “Procesos”, Gestión de Cambios, http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/gestion_de_cambios/proceso_gestion_de_cambios/proceso_gestion_de_cambios.php (25 noviembre 2010).

En la figura 14 se muestra el proceso que debe seguir un RFC utilizando la gestión de cambios de ITIL.

Figura 14. **Proceso para realizar un RFC de ITIL**



Fuente: osiatis. Gestión de cambios. http://itil.osiatis.es/Curso_ITIL/Gestión_Servicios_TI/gestión_de_cambios/visión_general_gestión_de_cambios/visión_general_gestión_de_cambios.php. Fecha de consulta: 26 de junio 2010.

5.4.2.1. Registro

"El primer paso del proceso de cambio es registrar las RFCs, no siempre un cambio implica una RFC, para cambios de poca importancia o que se repiten periódicamente se puede acordar un procedimiento estándar.

Un RFC requiere como mínimo los siguientes datos:

- Fecha de recepción
- Identificador único de la RFC
- Identificador del error conocido asociado
- Descripción del cambio propuesto
  - Motivación
  - Propósito
  - *Configuration ítems* CIs involucrados
  - Estimación de recursos necesarios para la implementación
  - Tiempo estimado
- Estatus: que inicialmente será el de "registrado"

Este registro deberá ser actualizado con toda la información generada durante el proceso para permitir un detallado seguimiento del mismo desde su aprobación hasta la evaluación final y cierre.

La información de registro debe ser actualizada durante todo el proceso y debe incluir al menos:

- Estatus actualizado: "aceptado", "rechazado", "implementado", etc.
- Fecha de aceptación (denegación) del RFC
- Evaluación preliminar de la gestión del cambio

- Prioridad y categoría
- Planes de "*back out*"
- Recursos asignados
- Fecha de implementación
- Plan de implementación
- Cronograma
- Revisión post-implementación
- Evaluación final
- Fecha de cierre"[23]

5.4.2.2. Aceptación y clasificación

"Luego del registro de la RFC se debe de evaluar si se acepta o se rechaza, la aceptación del RFC no implica su posterior aprobación por el CAB, tras la aceptación se debe de asignar al RFC una prioridad y categoría dependen de la urgencia y el impacto de la misma.

La categoría determina la dificultad e impacto de la RFC y será el parámetro relevante para determinar la asignación de recursos necesarios, los plazos previstos y el nivel de autorización requerido para la implementación del cambio.

El rango de prioridades puede ser definido de diferentes formas, cada empresa puede determinar las prioridades necesarias, pero como mínimo debe de incluir las siguientes:

[23]Osiatis. "Registro", Gestión de Cambios, http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/gestion_de_cambios/proceso_gestion_de_cambios/registro_del_cambio.php (25 noviembre 2010).

- Bajo
- Normal
- Alto
- Urgente

Un cambio menos no necesariamente necesita la aprobación del CAB y estos pueden ser implementados directamente".[24]

5.4.2.3. Aprobación y planificación

"El CAB debe de reunirse periódicamente para analizar los RFCs pendientes y elaborar el calendario de cambios.

Para su aprobación el cambio se debe evaluar minuciosamente:

- ¿Cuáles son los beneficios esperados del cambio propuesto?
- ¿Justifican esos beneficios los costes asociados al proceso de cambio?
- ¿Cuáles son los riesgos asociados?
- ¿Disponemos de los recursos necesarios para llevar a cabo el cambio con garantías de éxito?
- ¿Puede demorarse el cambio?
- ¿Cuál será el impacto general sobre la infraestructura y la calidad de los servicios TI?
- ¿Puede el cambio afectar los niveles establecidos de seguridad TI?

[24] Osiatis. "Aceptación y Clasificación", Gestión de Cambios, http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/gestion_de_cambios/proceso_gestion_de_cambios/aceptacion_y_clasificacion_del_cambio.php (25 noviembre 2010).

Una vez aprobado el cambio (en caso contrario se sigue el proceso ya descrito en la figura 14 para el caso de no aceptación) debe evaluarse si éste ha de ser implementado aisladamente o dentro de un "paquete de cambios" que formalmente equivale a un sólo cambio. Esto tiene algunas ventajas:

- Se optimizan los recursos necesarios.
- Se evitan posibles incompatibilidades entre diferentes cambios.
- Sólo se necesita un plan de *back-out*.
- Se simplifica el proceso de actualización de *la configuration management database* CMDB y la revisión post-implementación".[25]

5.4.2.4. Implementación

"La gestión de cambios no es la encargada de realizar el cambio, pero sí de supervisar y coordinar todo el proceso; en la fase de desarrollo del cambio se deberá monitorizar el proceso para asegurar que:

- Tanto el *software* desarrollado como el *hardware* adquirido se ajustan a las especificaciones predeterminadas.
- Se cumplen los calendarios previstos y la asignación de recursos es la adecuada.
- El entorno de pruebas es realista y simula adecuadamente el entorno de producción.
- Los planes de "*back-out"* permiten la rápida recuperación de la última configuración estable.

[25]Osiatis. "Aprobación y Planificación", Gestión de Cambios, http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/gestion_de_cambios/proceso_gestion_de_cambios/planificacion_del_cambio.php (25 noviembre 2010).

Si es posible, debe permitirse el acceso restringido de usuarios al entorno de pruebas para que realicen una valoración preliminar de los nuevos sistemas en lo que respecta a su:

- Funcionalidad
- Usabilidad
- Accesibilidad

La opinión de los usuarios debe de ser tomada en cuenta y la RFC debe de ser revisado si existe una objeción justificada al cambio".[26]

5.4.2.5. Evaluación

"Antes de realizar el cierre al cambio es conveniente realizar una evaluación que permita validar el impacto del mismo en la calidad del servicio y en la productividad.

Los aspectos fundamentales a tener en cuenta son:

- ¿Se cumplieron los objetivos previstos?
- En qué medida se apartó el proceso de las previsiones realizadas por la gestión de cambios.
- ¿Provocó el cambio problemas o interrupciones del servicio imprevisto?
- ¿Cuál ha sido la percepción de los usuarios respecto al cambio?
- ¿Se pusieron en marcha los planes de "*back-out*" en alguna fase del proceso? ¿Por qué?

[26]Osiatis. "Implementación", Gestión de Cambios, http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/gestion_de_cambios/proceso_gestion_de_cambios/implementacion_del_cambio.php(25 noviembre 2010).

Si la evaluación final determina que el cambio es satisfactorio se procederá al cierre del RFC".[27]

5.4.2.6. Emergencias

"Los cambios de emergencia son resultado de una planificación deficiente y a veces resultan inevitables, el procedimiento a seguir en estos casos debe estar previsto, de establecer un protocolo de evaluación de los cambios urgentes, como por ejemplo: una reunión urgente del CAB y/o comité de emergencia, una decisión del gestor de cambios si es imposible demorar la resolución del problema, etc".[28]

[27]Osiatis. "Evaluación", Gestión de Cambios, http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/gestion_de_cambios/proceso_gestion_de_cambios/evaluacion_del_cambio.php(25 noviembre 2010).
[28]Osiatis. "Cambios de Emergencia", Gestión de Cambios, http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/gestion_de_cambios/proceso_gestion_de_cambios/cambios_de_emergencia.php (25 noviembre 2010).

Tabla XXI. **Gestión de cambios, Municipalidad de Guatemala**

Departamento de TI / **Factor gestión de cambios**	**Informática Municipal**	**Emetra**	**Empagua**	**Licencias de la construcción**	**Catastro**
Estado actual	Utiliza un *software* para llevar el control de los cambios.	El control de los cambios lo llevan los desarrolladores.	El control de los cambios lo llevan los desarrolladores.	El control de los cambios lo llevan los desarrolladores.	El control de los cambios lo llevan los desarrolladores.
Problemas del estado actual	Los desarrolladores se les olvida subir el cambio al sistema de control de cambios. Tiempo para realizar el cambio lo define el desarrollador en base a su conocimiento. No existe un proceso estándar para realizar los cambios.	Cuando existe un cambio de personal y es necesario realizar una actualización al *software*, no se tiene conocimiento de cuál es la última versión. La priorización de los cambios se realiza en base al nivel organizativo. El tiempo para realizar el cambio lo define el desarrollador en base a sus conocimientos.	Cuando existe un cambio de personal y es necesario realizar una actualización al *software*, no se tiene conocimiento de cuál es la última versión. El tiempo para realizar el cambio lo define el desarrollador en base a sus conocimientos.	Cuando existe un cambio de personal y es necesario realizar una actualización al *software*, no se tiene conocimiento de cuál es la última versión. El tiempo para realizar el cambio lo define el desarrollador en base a sus conocimientos.	Cuando existe un cambio de personal y es necesario realizar una actualización al *software*, no se tiene conocimiento de cuál es la última versión. El tiempo para realizar el cambio lo define el desarrollador en base a sus conocimientos.
Beneficios de implementación	Verificar si el cambio realizado se encuentra en el software de cambios, antes de colocar el nuevo cambio en producción. Analizar el impacto del cambio para definir los tiempos. Definir un proceso para la gestión de cambios.	Ayuda a tener el control de los cambios realizados y si existe cambio de personal el departamento no tendrán problemas porque ya se tendrá controlada la gestión de los cambios. Analizar el impacto del cambio para definir prioridades y tiempos. Definir un proceso para la gestión de cambios.	Ayuda a tener el control de los cambios realizados y si existe cambio de personal el departamento no tendrán problemas porque ya se tendrá controlada la gestión de los cambios. Analizar el impacto del cambio para definir prioridades y tiempos. Definir un proceso para la gestión de cambios.	Ayuda a tener el control de los cambios realizados y si existe cambio de personal el departamento no tendrán problemas porque ya se tendrá controlada la gestión de los cambios. Analizar el impacto del cambio para definir prioridades y tiempos, Definir un proceso para la gestión de cambios.	Ayuda a tener el control de los cambios realizados y si existe cambio de personal el departamento no tendrán problemas porque ya se tendrá controlada la gestión de los cambios. Analizar el impacto del cambio para definir prioridades y tiempos. Definir un proceso para la gestión de cambios.

Fuente: elaboración propia.

5.5. Satisfacción del cliente

Muchas de las empresas que desarrollan sistemas no toman en cuenta una medición sobre la satisfacción de los clientes, esto es importante debido a que permite una retroalimentación sobre qué puntos se pueden mejorar para realizar sistemas futuros. Adicionalmente obtenemos la percepción del cliente sobre el sistema, es decir si las expectativas del cliente fueron alcanzadas o sobrepasadas.

Para poder lograr la satisfacción de los clientes no se debe olvidar algunos de los principales factores que son: la calidad y el servicio que se le brinda, cómo medir la satisfacción del cliente, es muy fácil, ya que se les pregunta, esto se puede hacer mediante herramientas, las herramientas permiten utilizar dos tipos de datos, cualitativos y cuantitativos; la diferencia entre ambos tipos reside en que los datos cualitativos pretende entender de manera subjetiva la experiencia del cliente, éste se realiza a través de entrevistas, observaciones, etc., los datos cualitativos son medibles debido a que se recolectan utilizando un estándar, como por ejemplo: una escala numérica (1 = Pésimo …. 5 = Excelente), se puede realizar usando encuestas.

5.5.1. Cómo crear una encuesta para medir la satisfacción del cliente

Es importante como saber realizar una encuesta para medir la satisfacción de los clientes, no es la única opción para medir la satisfacción de los clientes, pero es un método que ayuda a obtener el resultado que se desea.

Los pasos que debemos de seguir para obtener un resultado satisfactorio y que cumpla con nuestras necesidades son los siguientes:

- Objetivo de la encuesta
- Escala de medición
- Número y tipo de preguntas
- Prueba piloto

5.5.1.1. Objetivo de la encuesta

Se identifica qué tipo de información se quiere obtener del cliente, por ejemplo: uso del sistema, tiempo de respuesta, etc., este es el primer paso y el más importante, muchos no realizan este paso y el problema surge cuando se están analizando los datos.

5.5.1.2. Escala de medición

Es importante definir una escala de medición de la encuesta, el error que se comete cuando se realiza una encuesta es que se definen las preguntas de forma abierta, es decir el cliente coloca con sus palabras la respuesta y esto provoca que no se pueda realizar un análisis de la encuesta.

Existen varias formas de definir una escala de medición, cada una de las empresas que desea desarrollar una encuesta puede definir su propia escala para medir la satisfacción del cliente, por ejemplo: si se desea medir el tiempo de respuesta del sistema se coloca una escala de 1 a 3 donde 1 es lento, 2 normal y 3 rápido.

5.5.1.3. Número y tipo de preguntas

No existe un número determinado de preguntas para una encuesta, pero sin embargo es importante que la encuesta recabe la información necesaria para cumplir con el objetivo, también es importante que no se extralimite con el número de preguntas porque el cliente puede perder el interés, si la encuesta es demasiado grande y al final se obtiene información no precisa sobre la satisfacción del cliente.

5.5.1.4. Prueba piloto

Es necesario realizar pruebas con la encuesta, éstas las puedes realizar con el equipo de trabajo, amigos, etc., el fin de las pruebas es identificar las preguntas que pudiesen no estar claras y entendibles, así como la distribución de las preguntas en la encuesta.

Tabla XXII. **Satisfacción del cliente, Municipalidad de Guatemala**

Departamento de TI	Informática Municipal	Emetra	Empagua	Licencias de la construcción	Catastro
Factor satisfacción del cliente					
Estado actual	No verifican la satisfacción de los clientes.	No verifican la satisfacción de los clientes.	No verifican la satisfacción de los clientes.	No verifican la satisfacción de los clientes.	No verifican la satisfacción de los clientes.
Problemas del estado actual	No se tiene la retroalimentación de los usuarios.	No se tiene la retroalimentación de los usuarios.	No se tiene la retroalimentación de los usuarios.	No se tiene la retroalimentación de los usuarios.	No se tiene la retroalimentación de los usuarios.
Beneficios de implementación	Retroalimentación de los usuarios para mejorar el desempeño de los desarrolladores. Tener estadísticas de usuarios satisfechos vs. usuarios insatisfechos.	Retroalimentación de los usuarios para mejorar el desempeño de los desarrolladores. Tener estadísticas de usuarios satisfechos vs. usuarios insatisfechos.	Retroalimentación de los usuarios para mejorar el desempeño de los desarrolladores. Tener estadísticas de usuarios satisfechos vs. usuarios insatisfechos.	Retroalimentación de los usuarios para mejorar el desempeño de los desarrolladores. Tener estadísticas de usuarios satisfechos vs. usuarios insatisfechos .	Retroalimentación de los usuarios para mejorar el desempeño de los desarrolladores. Tener estadísticas de usuarios satisfechos vs. usuarios insatisfechos.

Fuente: elaboración propia.

CONCLUSIONES

1. Los factores importantes para el desarrollo de un sistema en la Municipalidad de Guatemala son: reglas del negocio, metodologías para el modelado de procesos y actividades, toma de requerimientos, usuarios y roles, definición de estándares, tipos de pruebas, trabajo en equipo, manual de usuario, gestión de cambios.

2. Cada factor se tiene que realizar en diferentes etapas del desarrollo, existen factores que se pueden realizar en la misma etapa y además pueden estar relacionados.

3. Las empresas si desean realizar *software* de calidad deben estar dispuestas al cambio, se puede optar una nueva forma de desarrollo de sistemas informáticos utilizando los factores importantes.

4. La implementación de los factores para desarrollo de *software*, implica que las empresas tiene que invertir tiempo y en mucho de los casos dinero, si no se realiza la inversión no se garantiza el éxito del cambio.

5. Se presentó la influencia que ha tenido la implementación de alguno de los factores en los departamentos de TI de la Municipalidad de Guatemala.

RECOMENDACIONES

1. Es importante que los departamentos de TI de la Municipalidad de Guatemala tengan el conocimiento de los factores esenciales en el desarrollo de un *software*, esto les ayudará a tener un mayor porcentaje de éxito.

2. La implementación de algunos de los factores en los departamentos de TI en la Municipalidad de Guatemala, les ayudó a tener mayor éxito en el desarrollo de *software*, si se implementarán más factores en el desarrollo, obtendrían un mejor beneficio y adicional los usuarios tendrían un mejor servicio.

3. Utilizar los factores de manera inadecuada puede provocar el fracaso, es decir si realizamos un levantado de requerimientos pero no tiene la información relevante, de nada sirve la utilización de este factor.

4. No se debe abusar de la utilización de los factores de desarrollo, la Municipalidad de Guatemala debe de analizar y tomar los factores que crea necesarios para el desarrollo de los sistemas.

BIBLIOGRAFÍA

1. AGARWAL, B. B. *Software engineering & testing. United States of America: Jones and Bartlett Publishers*, 2010. 493 p.

2. BORREGO, Daniel. Como elaborar una encuesta para medir la satisfacción del cliente [en línea]. 20 de marzo de 2009. Disponible en Web: http://www.herramientasparapymes.com/como-elaborar-una-encuestas-para-medir-la-satisfacción-del-cliente.

3. ———.Como medir la satisfacción del cliente [en línea]. 11 de febrero de 2009. Disponible en Web: http://www.herramientasparapymes.com/%C2%BFcomo-medir-la-satisfaccion-del-cliente-parte-i.

4. Consultores Aiteco [en línea]. España, [fecha de consulta: 2 mayo 2010]. Disponible en Web: http://www.aiteco.com/modelpro.htm.

5. Dedgerencia.com [en línea]. Meltom *Technologies* Inc. [fecha de consulta: 26 mayo 2010]. Disponible en Web: http://www.degerencia.com/tema/trabajo_en_equipo.

6. ITIL. Glosario de términos ITIL [en línea]. 1 de mayo de 2006. Disponible en Web: http://www.itil-officialsite.com/nmsruntime/saveasdialog.asp? lID=925&sID=242.

7. *KATZENBACH*, *Jon* R. *El Trabajo en equipo ventajas y dificultades*. España: GRANICA, 2000. 355 p.

8. LEÓN SERRANO, Gonzalo*. Ingeniería de sistemas de software*. Madrid: Isdefe, 1996. 215 p.

9. Osiatis. [en línea]. España. [fecha de consulta: 26 junio 2010]. Disponible en Web: http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/gestion_de_cambios/proceso_gestion_de_cambios/ proceso_ gestion_de_cambios.php.

10. *PRESSMAN*, *Roger* S. *Ingeniería del software un enfoque práctico*. 5a ed. España: McGraw-Hill, 2002. 589 p.

11. *Software* Gurú [en línea]. Marzo - abril 2007. México. Disponible en Web: http://www.sg.com.mx/content/view/271.

12. Wikipedia [en línea]. *United States*, [fecha de consulta: 5 junio 2010]. Disponible en Web: http://en.wikipedia.org/wiki/Work_system.

13. Wikipedia [en línea]. *United States*, [fecha de consulta: 7 julio 2010]. Disponible en Web: http://es.wikipedia.org/wiki/Requerimiento_(sistemas).

14. Wikipedia [en línea]. *United States*, [fecha de consulta: 8 julio 2010]. Disponible en Web: http://es.wikipedia.org/wiki/Contrato.

15. Wikipedia [en línea]. *United States*, [fecha de consulta: 28 abril 2010]. Disponible en Web: http://es.wikipedia.org/wiki/Reglas_de_negocio.

16. *York*, *University* [en línea]. Mediawiki, [fecha de consulta: 5 junio 2010]. Disponible en Web: http://www.fsc.yorku.ca/york/istheory/wiki/index.php/Work_systems_theory.